



**Università
degli Studi
di Palermo**



Introduzione ai Calcolatori

CALCOLATORI ELETTRONICI – FONDAMENTI DI PROGRAMMAZIONE

a.a. 2023/2024

Prof. Roberto Pirrone

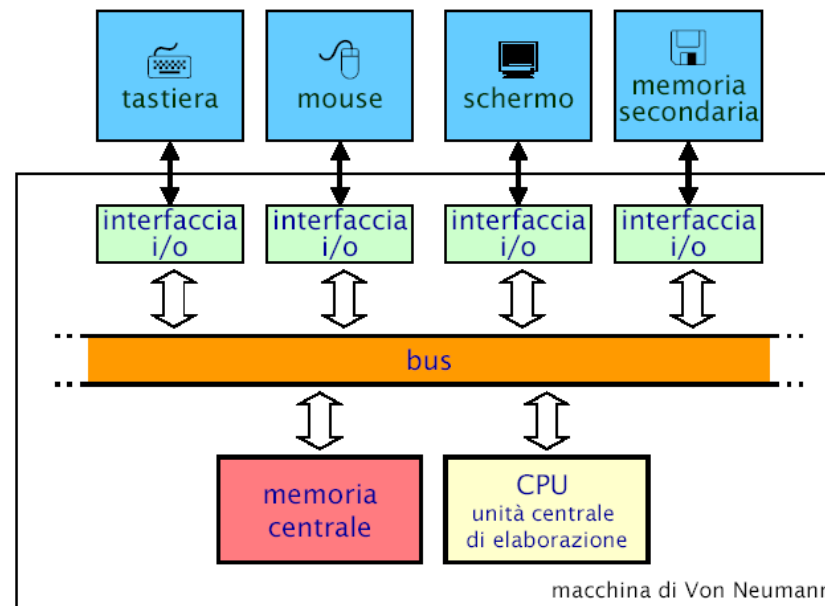
Sommario

- Architettura del Calcolatore
- Sistemi Operativi
- Linguaggi di Programmazione
- Traduzione dei programmi



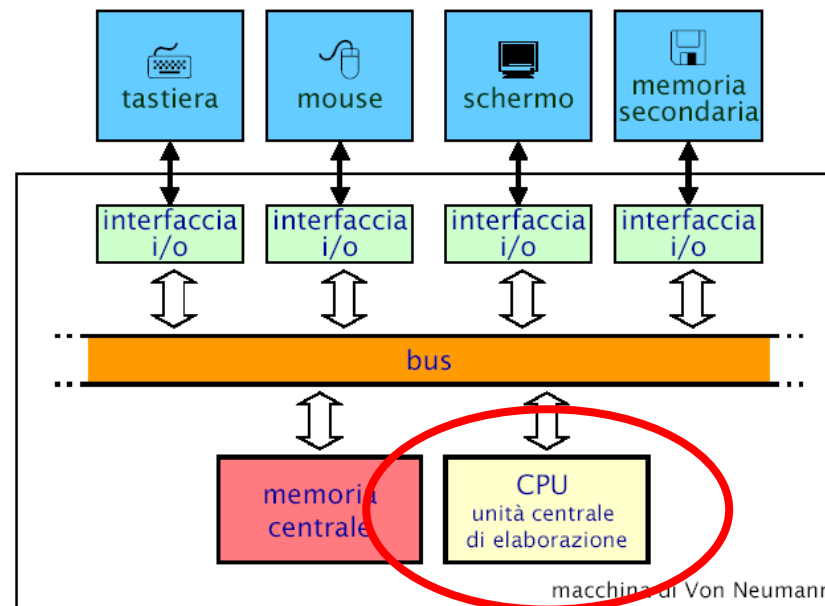
Architettura del Calcolatore

- Architettura a “programma memorizzato”
 - Dati e istruzioni dei programmi vanno opportunamente codificati come bit e **memorizzati** nello stesso luogo
 - Detta anche “Macchina” di Von Neumann che realizzò così il primo calcolatore elettronico secondo questo paradigma



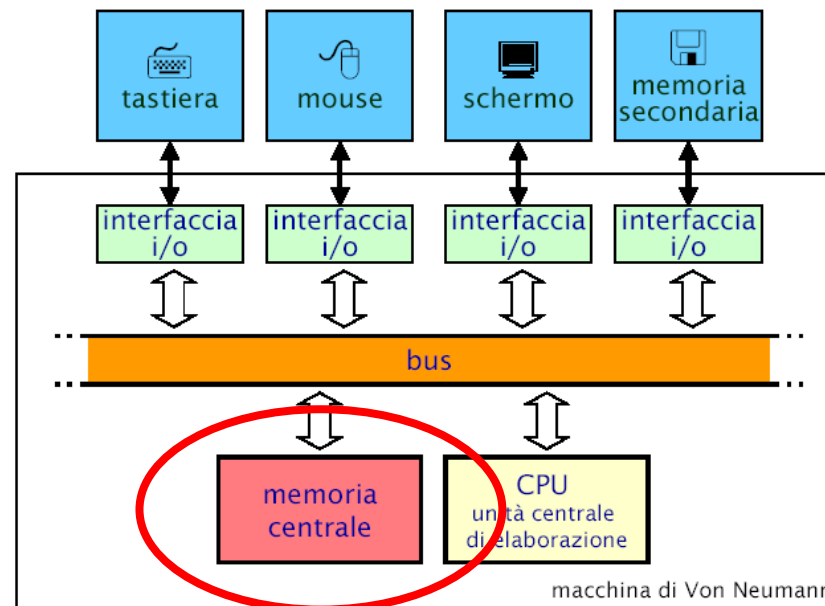
Architettura del Calcolatore

- CPU (Central Processing Unit), o Processore
 - Svolge le elaborazioni e il trasferimento dei dati, cioè esegue i programmi.
 - Svolge anche le funzioni di controllo dell'esecuzione
 - Ogni istruzione viene processata allo stesso modo, secondo il **ciclo-macchina**:
prelevamento → decodifica → esecuzione



Architettura del Calcolatore

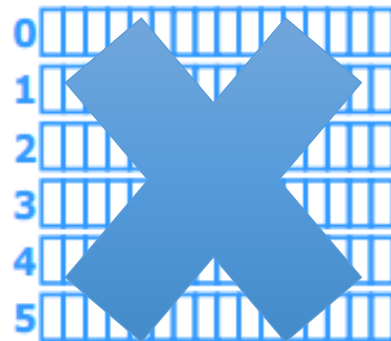
- Memoria centrale
 - RAM (Random Access Memory) è volatile (perde il suo contenuto quando si spegne il calcolatore) ed è usata per memorizzare dati e programmi.
 - E' organizzata in **celle**, le unità minime indirizzabili, ciascuna accessibile tramite un **indirizzo**



Architettura del Calcolatore

Organizzazione della memoria

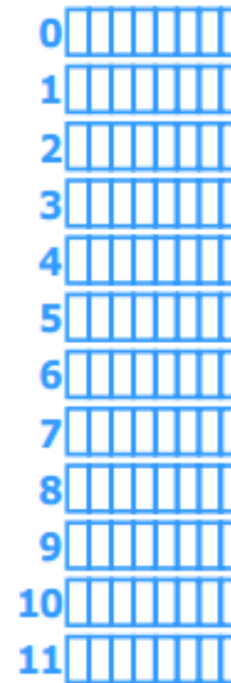
Una memoria può essere organizzata in diversi modi: con 96 bit possiamo avere 6 celle di 16 bit ($6 \cdot 16 = 96$), o 8 celle di 12 bit ($8 \cdot 12 = 96$) o 12 celle di 8 bit ($12 \cdot 8 = 96$).



6 parole da 16 bit



8 parole da 12 bit



12 parole da 8 bit

Architettura del Calcolatore

Organizzazione della memoria

- Le celle si organizzano in *parole* da 2, 4 o 8 byte ciascuna
- Parola*: blocco di byte che l'hardware (CPU) può elaborare, trasmettere, conservare tutto insieme

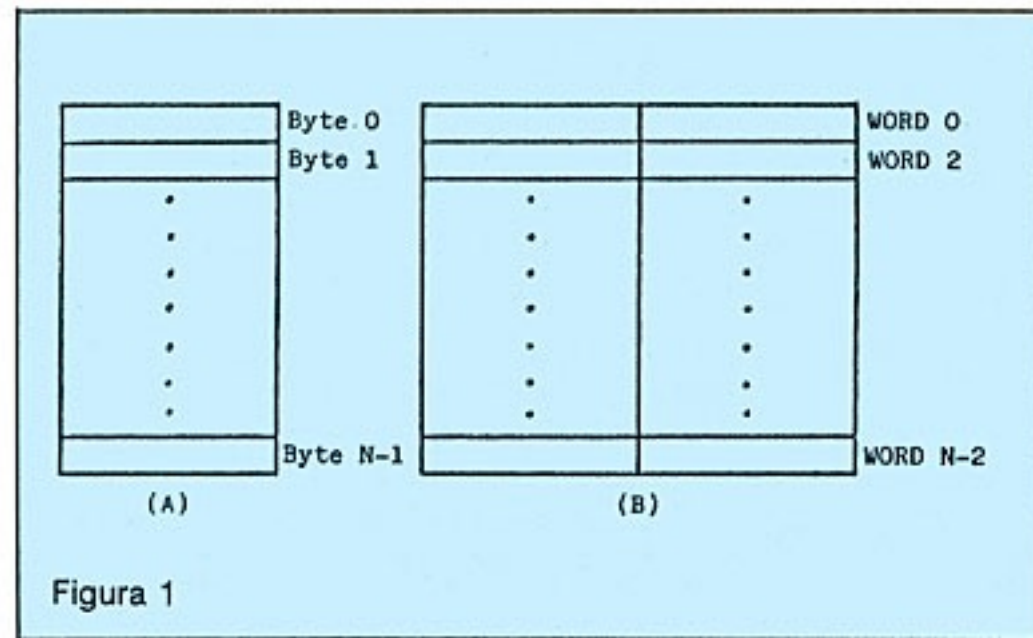
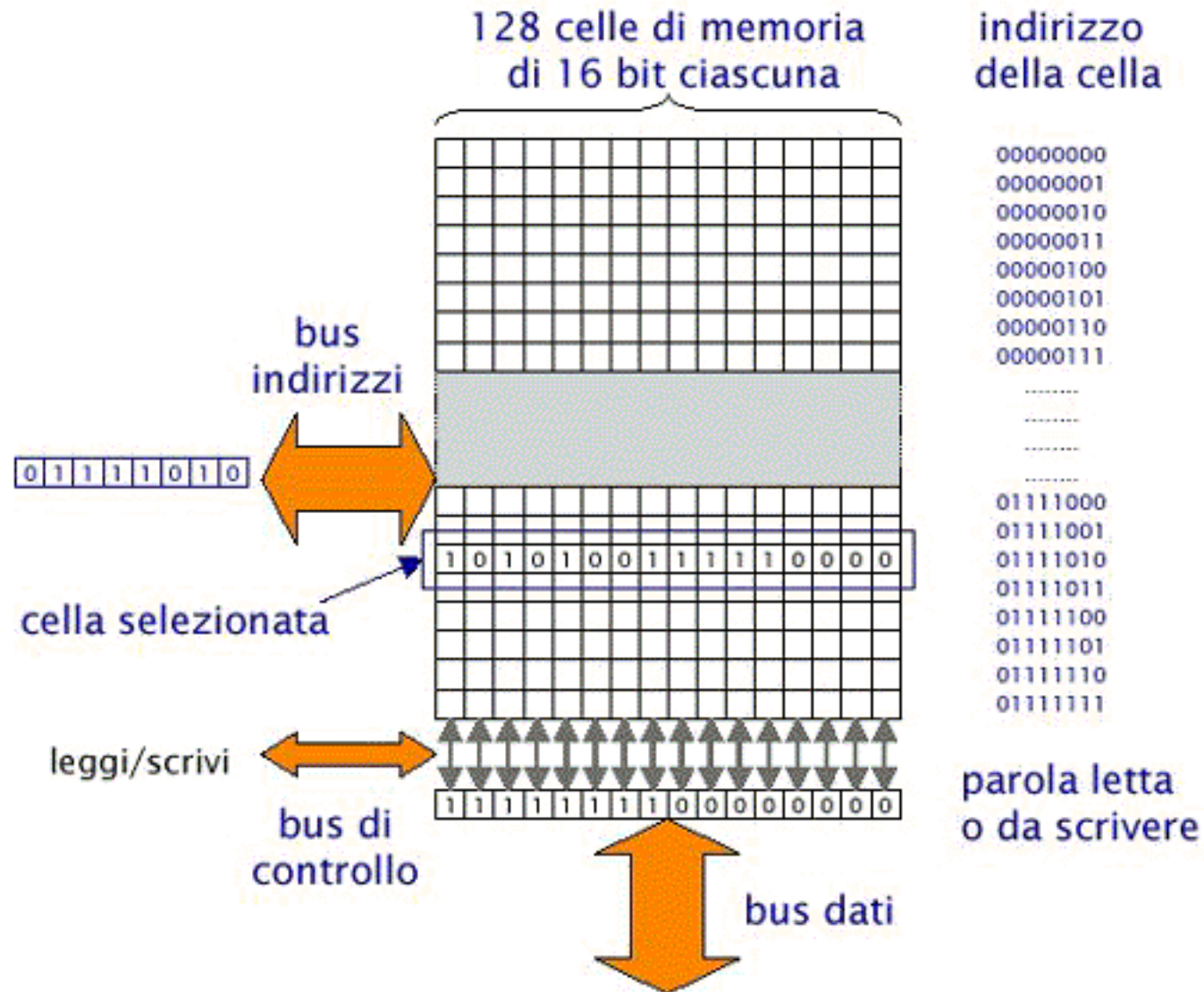


Figura 1 - Memoria organizzata a Byte (A) e a word di due Byte l'una (B).

Architettura del Calcolatore

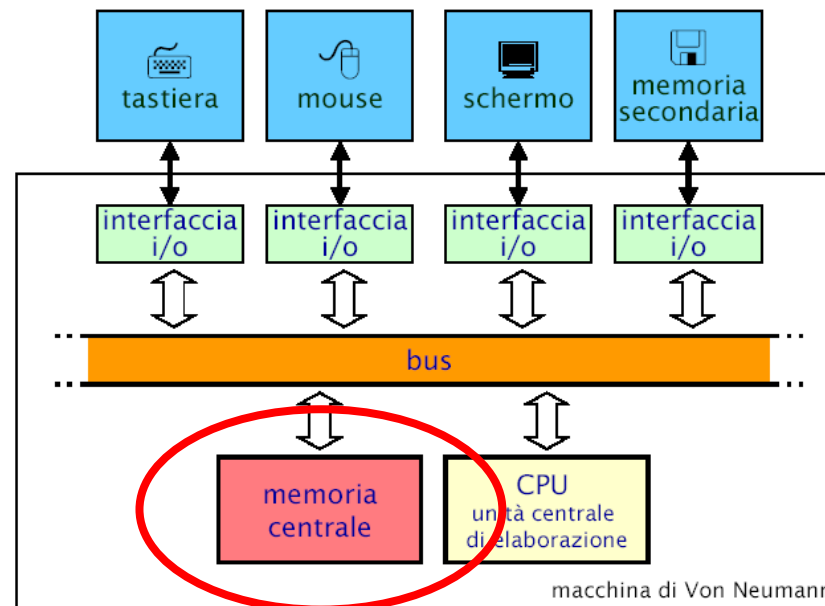
- Anche gli indirizzi della memoria sono rappresentati come numeri binari:
 - un indirizzo di M bit consente di indirizzare 2^M celle;
 - il numero di bit nell'indirizzo determina lo *spazio di indirizzamento*
 - il numero massimo di celle indirizzabili nella memoria
 - Quant'è lo spazio di indirizzamento di un calcolatore a 64 bit?

Architettura del Calcolatore



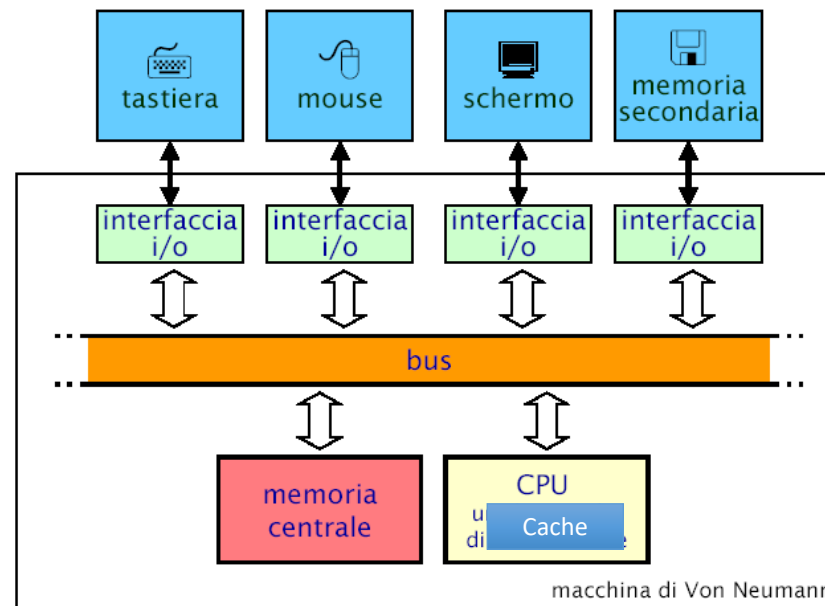
Architettura del Calcolatore

- Altre componenti della memoria centrale
 - *ROM (Read Only Memory)* è persistente (mantiene il suo contenuto quando si spegne il calcolatore) ma il suo contenuto è fisso e immutabile. È usata per memorizzare programmi di sistema



Architettura del Calcolatore

- Altre componenti della memoria centrale
 - *Cache* - Memoria di appoggio del processore, velocissima
 - Dimensioni relativamente limitate
 - Accesso estremamente rapido



Architettura del Calcolatore

- Memoria Cache
- Principio di località:
 - Quando il processore utilizza un'istruzione o dato è molto probabile che usi anche quelli ad esso vicini nella memoria (località spaziale).
 - Quando il processore utilizza un'istruzione o dato è molto probabile che lo usi di nuovo in breve tempo (località temporale).

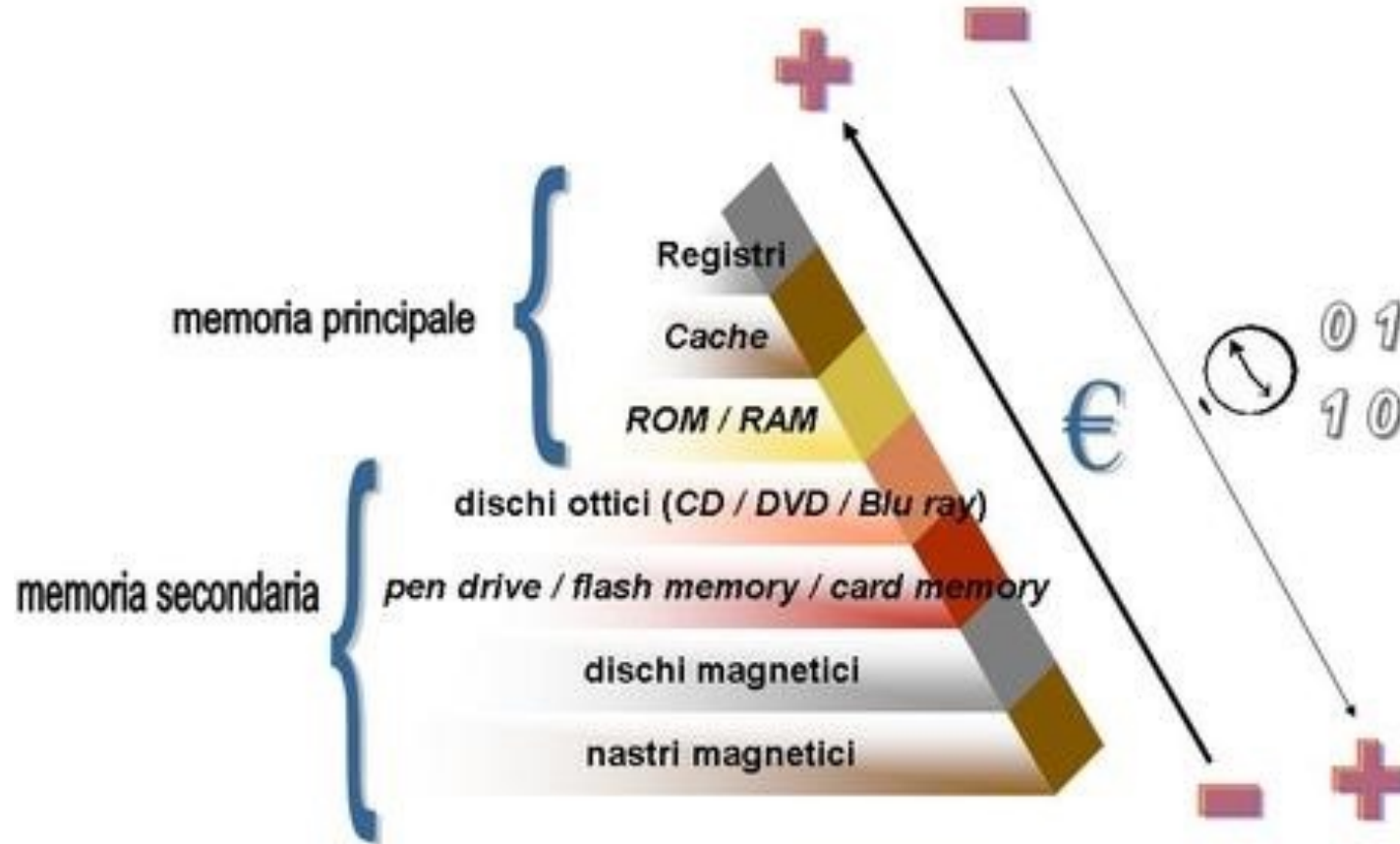
Architettura del Calcolatore

- Memoria Cache
- Il principio di località ispira il funzionamento della cache
 - Se viene richiesto un dato/istruzione viene prelevato il blocco di dati/istruzioni immediatamente vicini
 - Essi verranno conservati per un certo tempo e poi scartati se non più utilizzati.

Architettura del Calcolatore

- La soluzione ottimale per un sistema di memoria è:
 - Costo minimo
 - Capacità massima
 - Tempi di accesso minimi
- Soluzione approssimata: **GERARCHIA**
 - Tecnologie diverse possono soddisfare al meglio ciascuno dei requisiti. Una gerarchia cerca di ottimizzare globalmente i parametri.

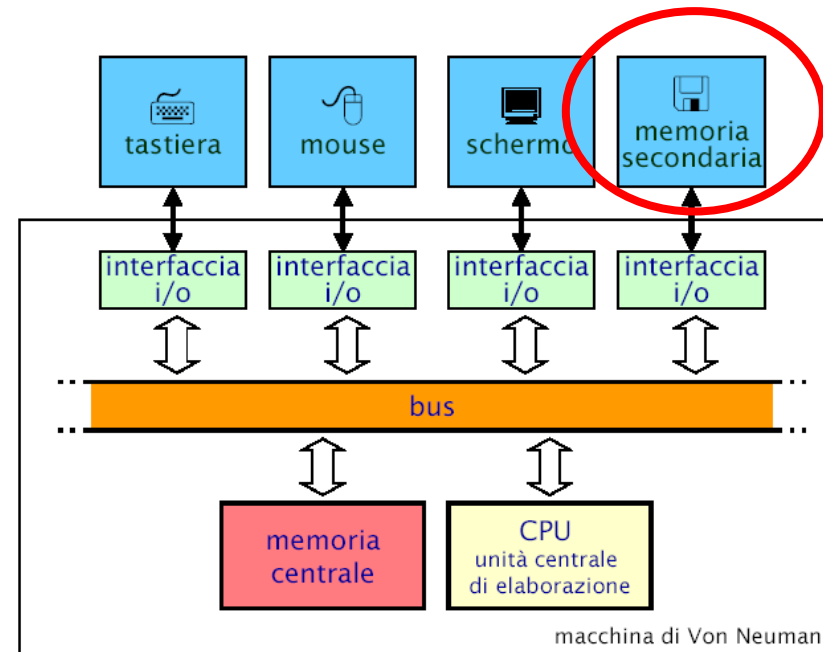
Architettura del Calcolatore



- La gerarchia della memoria fornisce l'illusione di una **memoria infinitamente grande e veloce.**

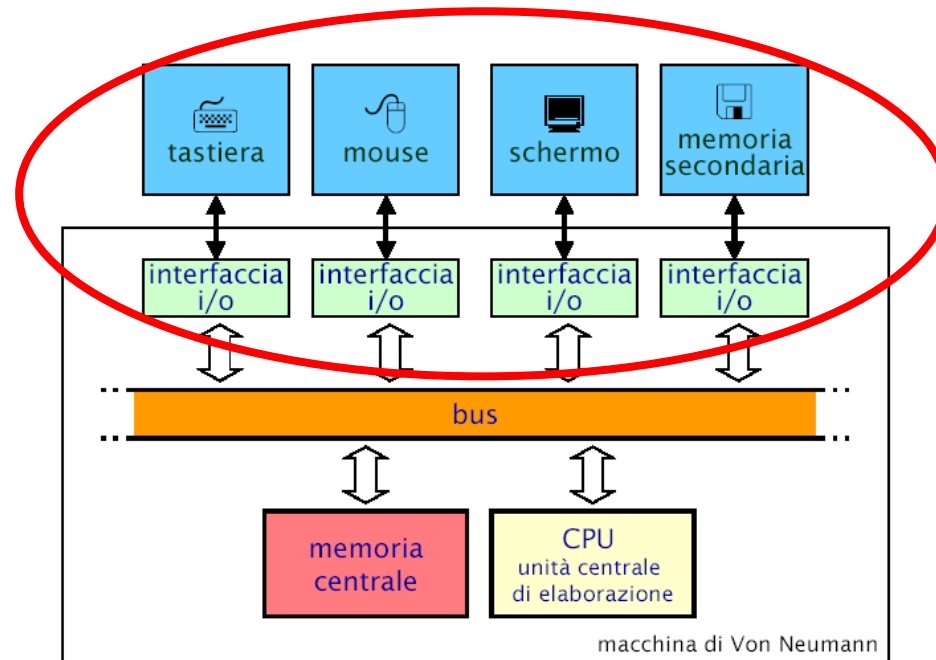
Architettura del Calcolatore

- Memoria secondaria (o di massa)
 - Dischi, nastri, CD, DVD
 - Memorizza grandi quantità di informazioni, ma è lenta: accesso in msec. contro nsec. della RAM – **rapporto di 10^6**
 - Persistente: Le informazioni non si perdono spegnendo la macchina



Architettura del Calcolatore

- Periferiche
 - Sono usate per far comunicare il calcolatore con l'esterno (in particolare con l'utente)
 - Non fanno parte della Macchina di Von Neumann, ma vi sono connesse attraverso le *interfacce di I/O*

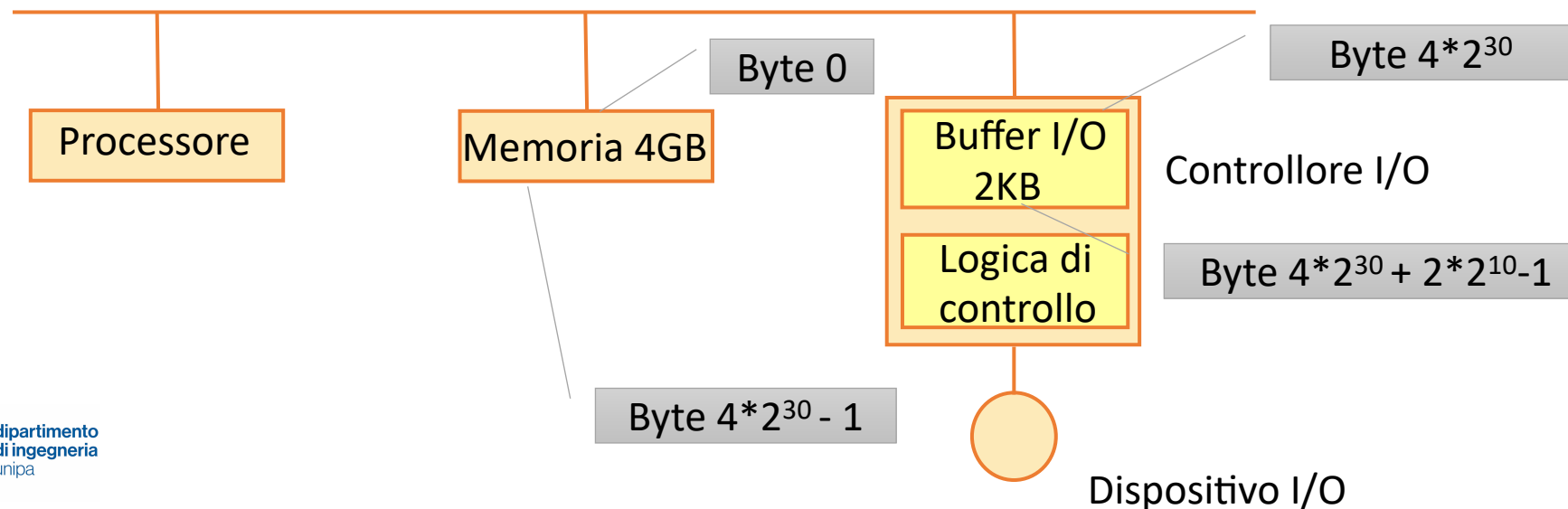


Architettura del Calcolatore

- Le periferiche sono molto più lente del processore e necessitano un modo per adeguare la propria velocità alla CPU
- Ogni periferica è connessa al bus tramite un *buffer* che è una piccola area di memoria RAM
- Il processore riversa i dati nel buffer alla propria velocità che è alta, mentre la periferica li preleva a bassa velocità e viceversa.

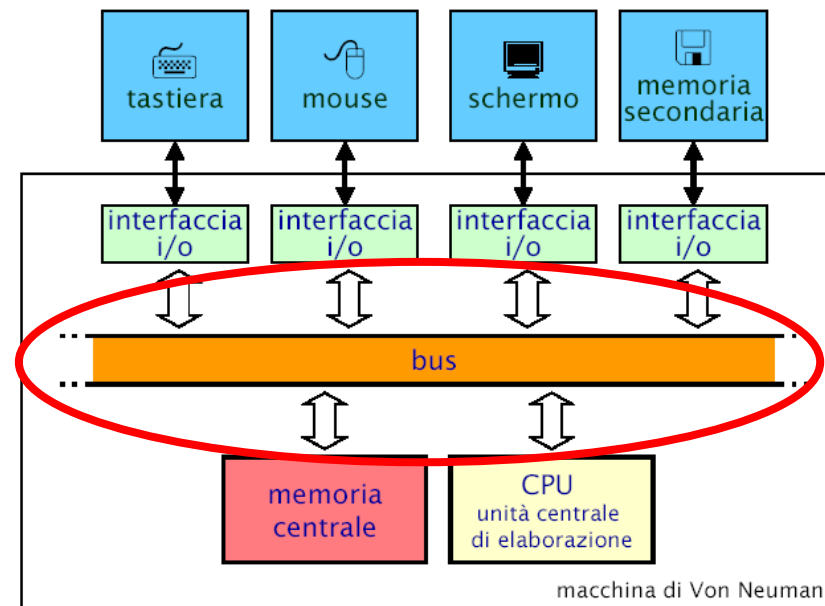
Buffer periferiche

- I buffer delle periferiche sono riferiti attraverso indirizzi di memoria successivi all'ultimo indirizzo RAM
- In genere, lo spazio di indirizzamento è molto più grande della dimensione della RAM ed è quindi possibile connettere molte periferiche



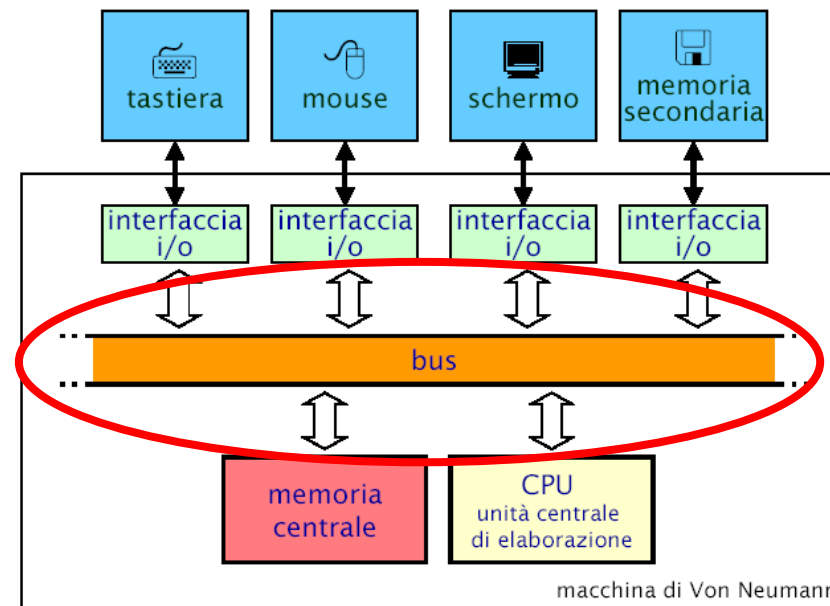
Architettura del Calcolatore

- Bus di sistema
 - Interconnette le componenti interne del calcolatore
- Collega due unità funzionali alla volta
 - una trasmette e l'altra riceve: funzionamento master/slave
 - la CPU (master) seleziona la connessione da attivare e ordina il trasferimento dei dati

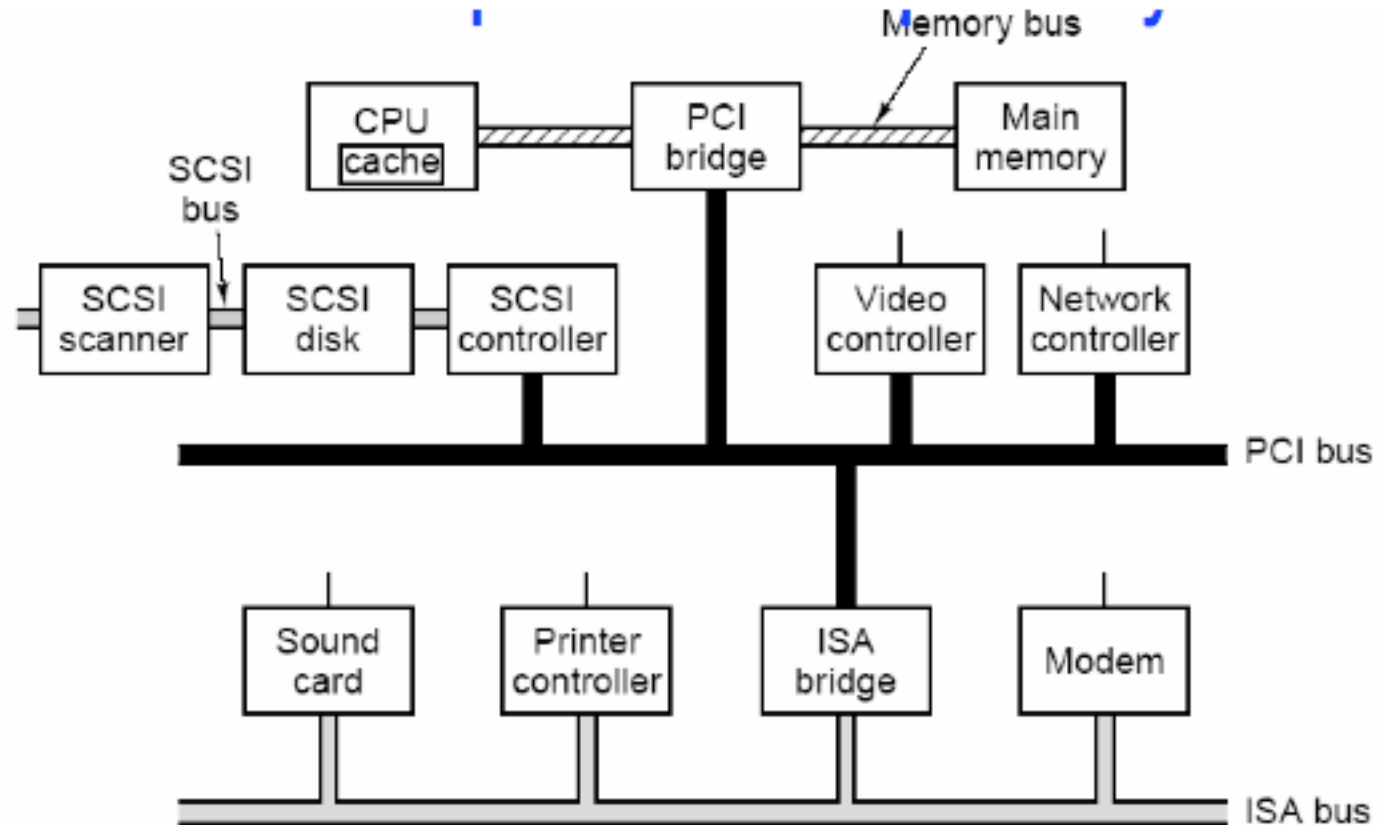


Architettura del Calcolatore

- Bus di sistema
 - Distinguiamo funzionalmente tre bus
 - *BUS DATI* (per istruzioni e dati dei programmi)
 - *BUS INDIRIZZI* (per accedere alle celle di memoria)
 - *BUS CONTROLLI* (per trasmettere i comandi)



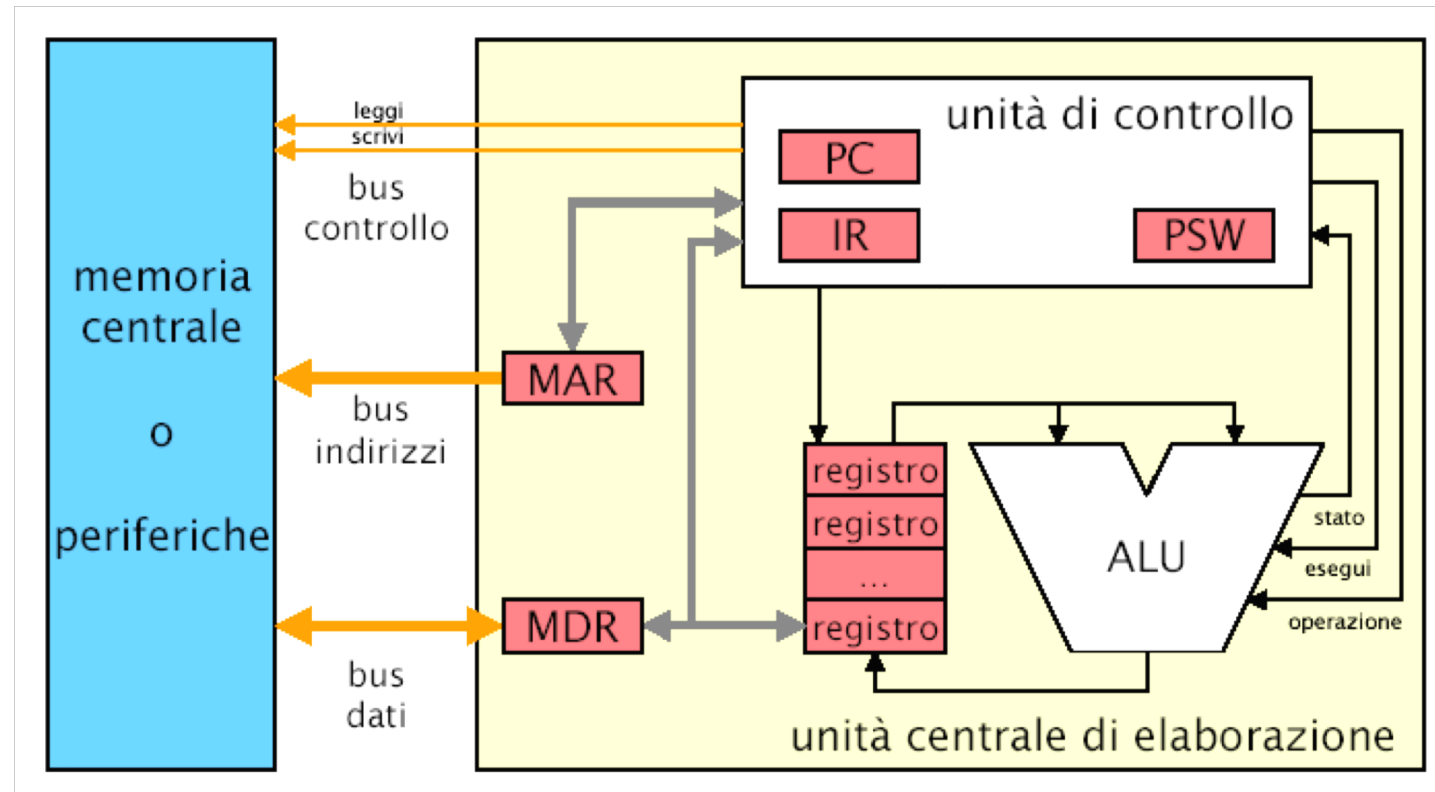
Architettura del Calcolatore



Architettura del Calcolatore

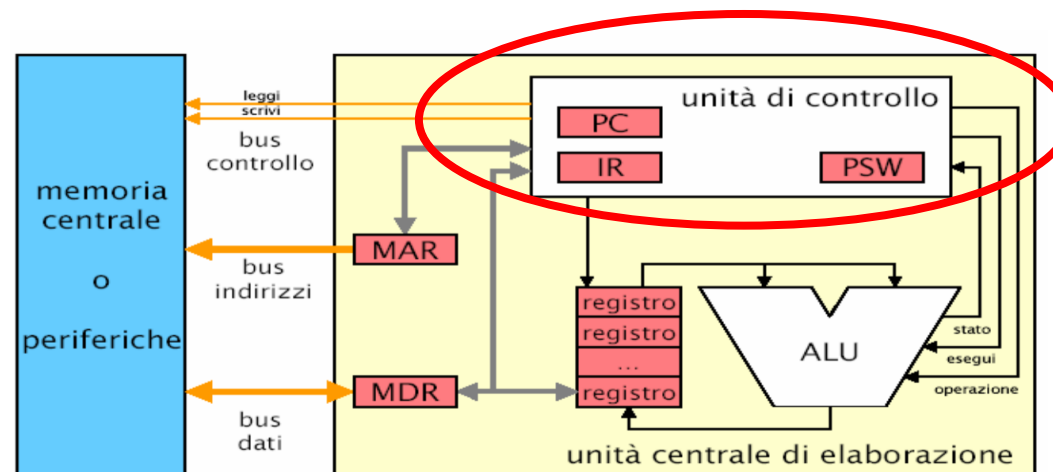
- CPU
 - Unità di controllo
 - Svolge funzioni di controllo, decide quali istruzioni eseguire.
 - Unità aritmetico logica (ALU)
 - esegue le operazioni aritmetico-logiche (+, -, *, / , confronti).
 - Registri
 - memoria ad alta velocità usata per risultati temporanei e informazioni di controllo;
 - il valore massimo memorizzabile in un registro è determinato dalle dimensioni del registro.

Architettura del Calcolatore



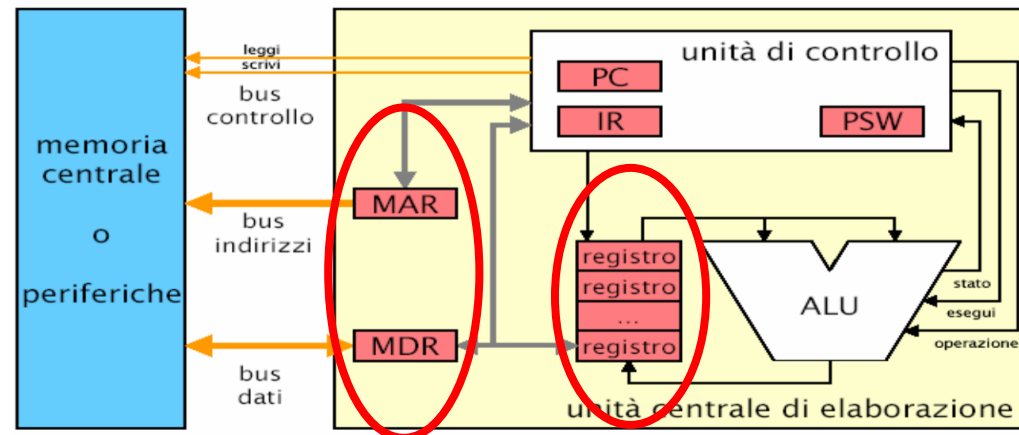
Architettura del Calcolatore

- Esistono registri di uso generico e registri specifici
 - **PC**: contatore delle istruzioni (Program Counter)
 - contiene l'indirizzo della prossima istruzione da eseguire
 - **IR**: registro delle istruzioni (Instruction Register)
 - contiene l'istruzione che deve essere eseguita
 - **PSW**: parola di stato del processore (Processor Status Word)
 - contiene informazioni, opportunamente codificate, sull'esito dell'ultima istruzione che è stata eseguita



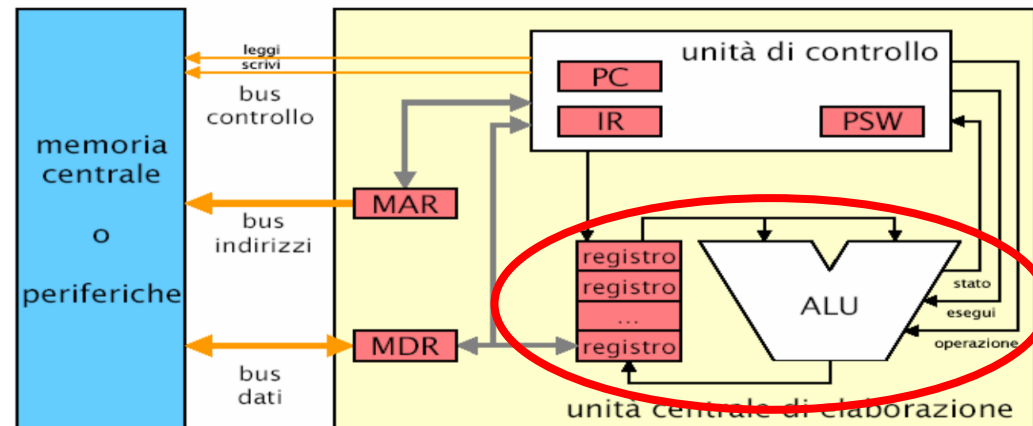
Architettura del Calcolatore

- Esistono registri di uso generico e registri specifici
 - **MAR**: registro indirizzi della memoria (Memory Address Register)
 - indirizzo della cella di memoria che deve essere acceduta o memorizzata
 - **MDR**: registro dati della memoria (Memory Data Register)
 - dato che è stato acceduto o che deve essere memorizzato
 - registri generali
 - per memorizzare gli operandi ed il risultato di una operazione

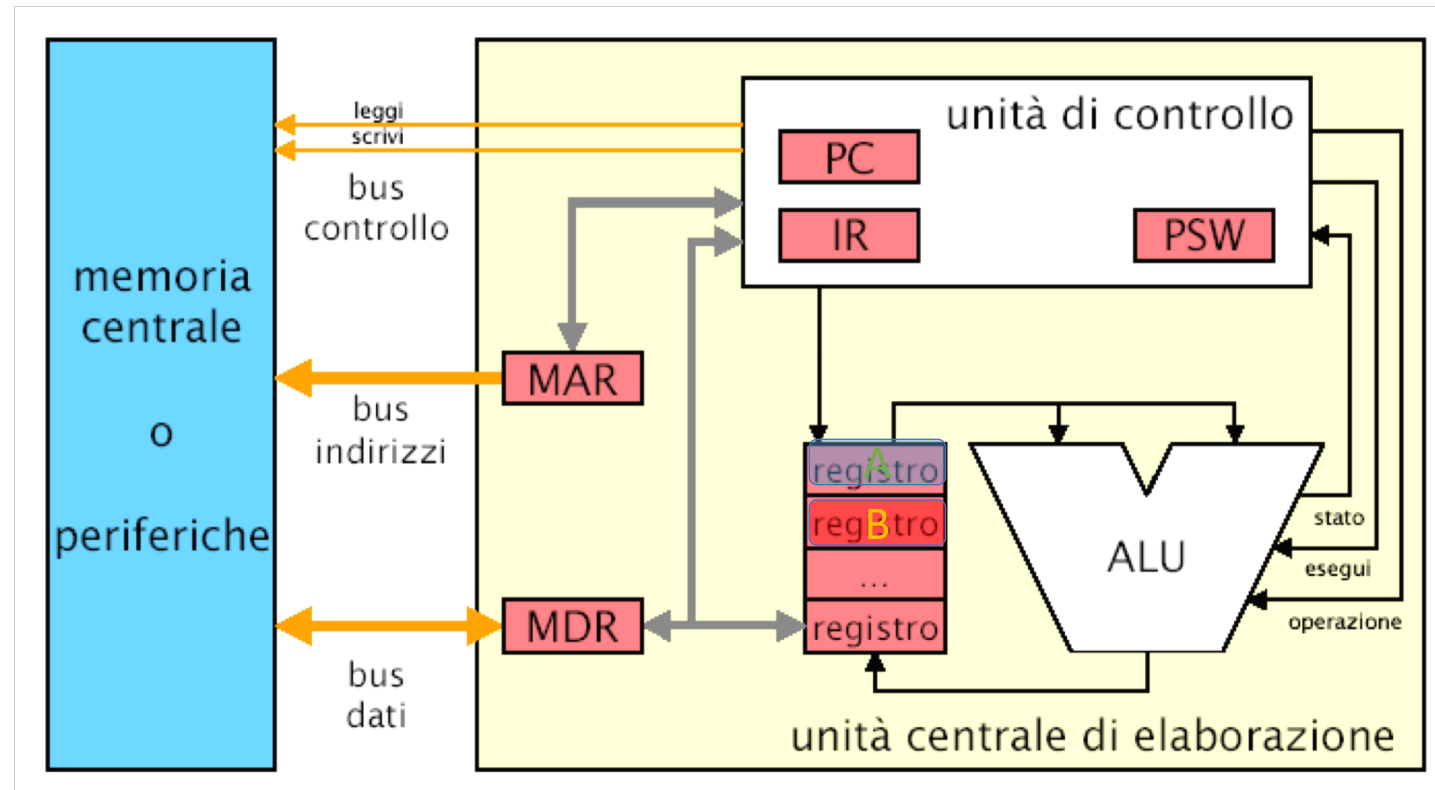


Architettura del Calcolatore

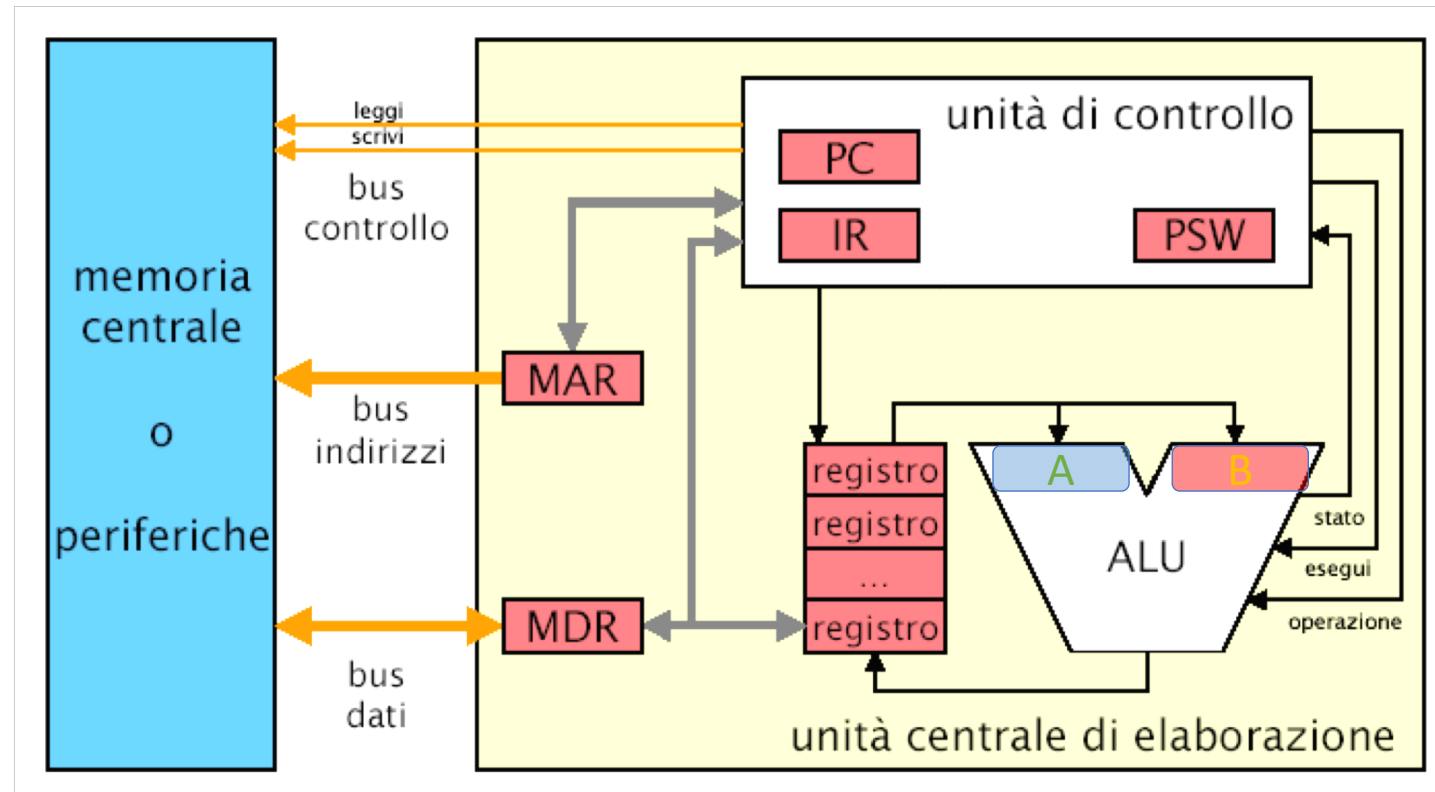
- L'Unità Aritmetico-Logica (ALU) è costituita da un insieme di circuiti in grado di svolgere le operazioni di tipo aritmetico e logico
- La ALU legge i valori presenti in alcuni registri, esegue le operazioni e memorizza il risultato in un altro registro



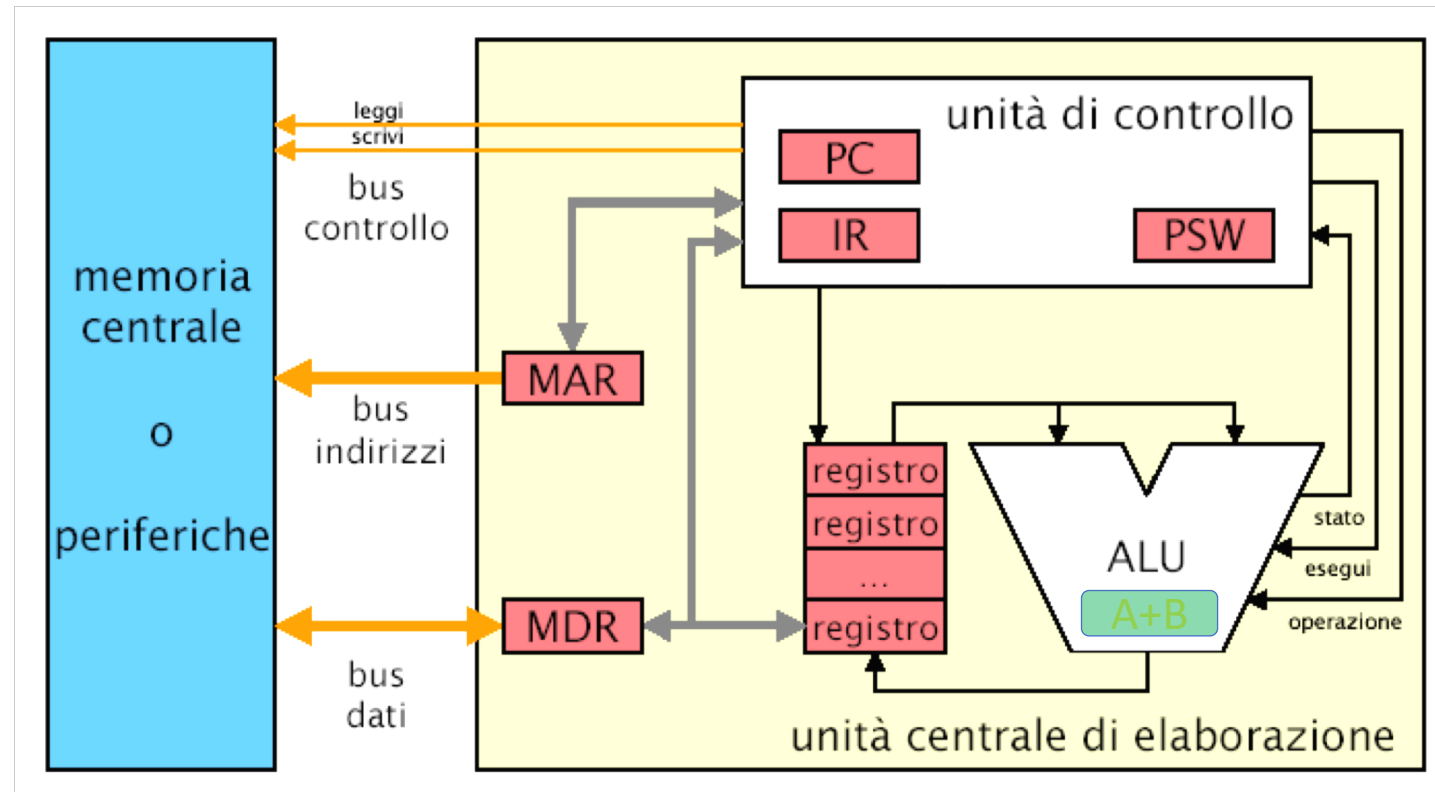
Architettura del Calcolatore



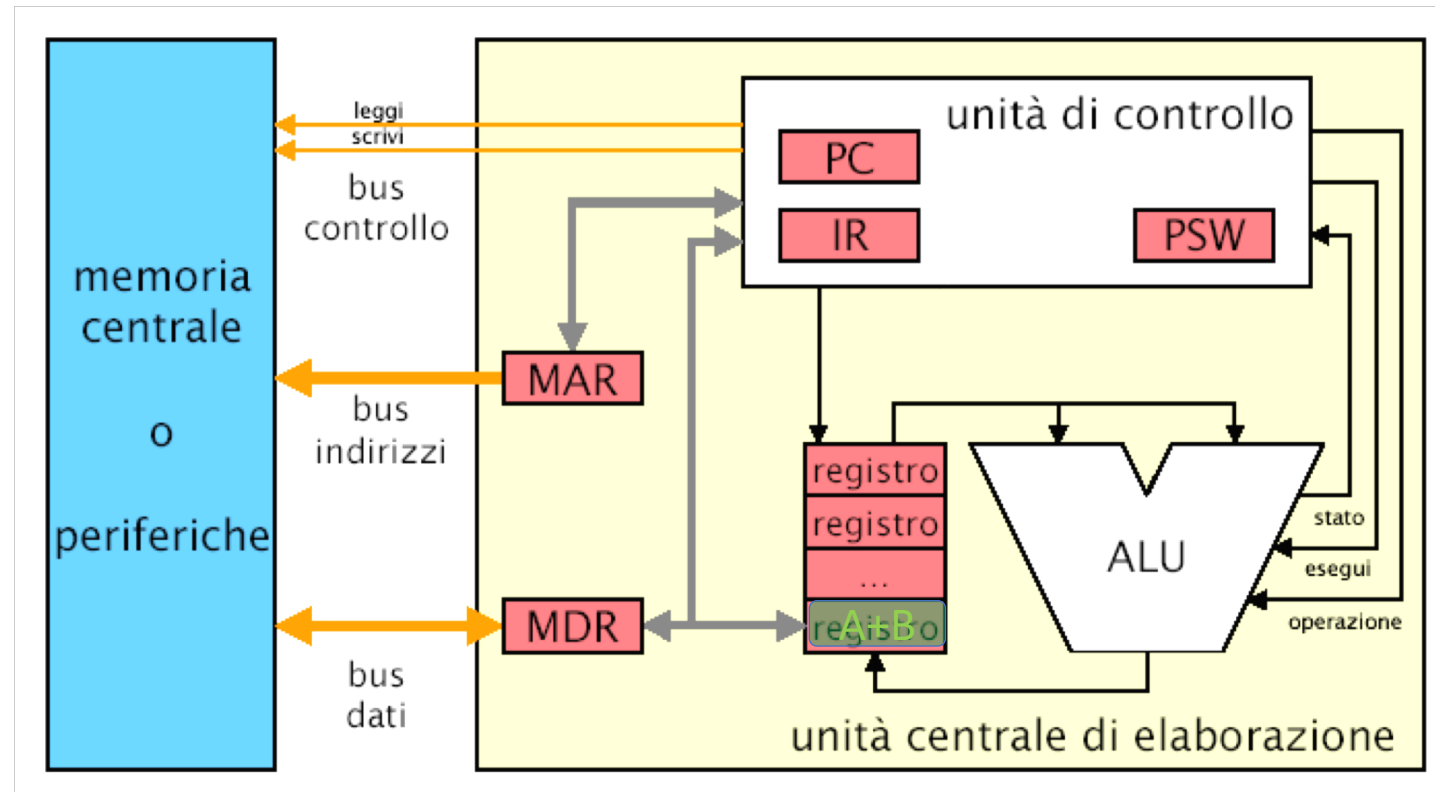
Architettura del Calcolatore



Architettura del Calcolatore



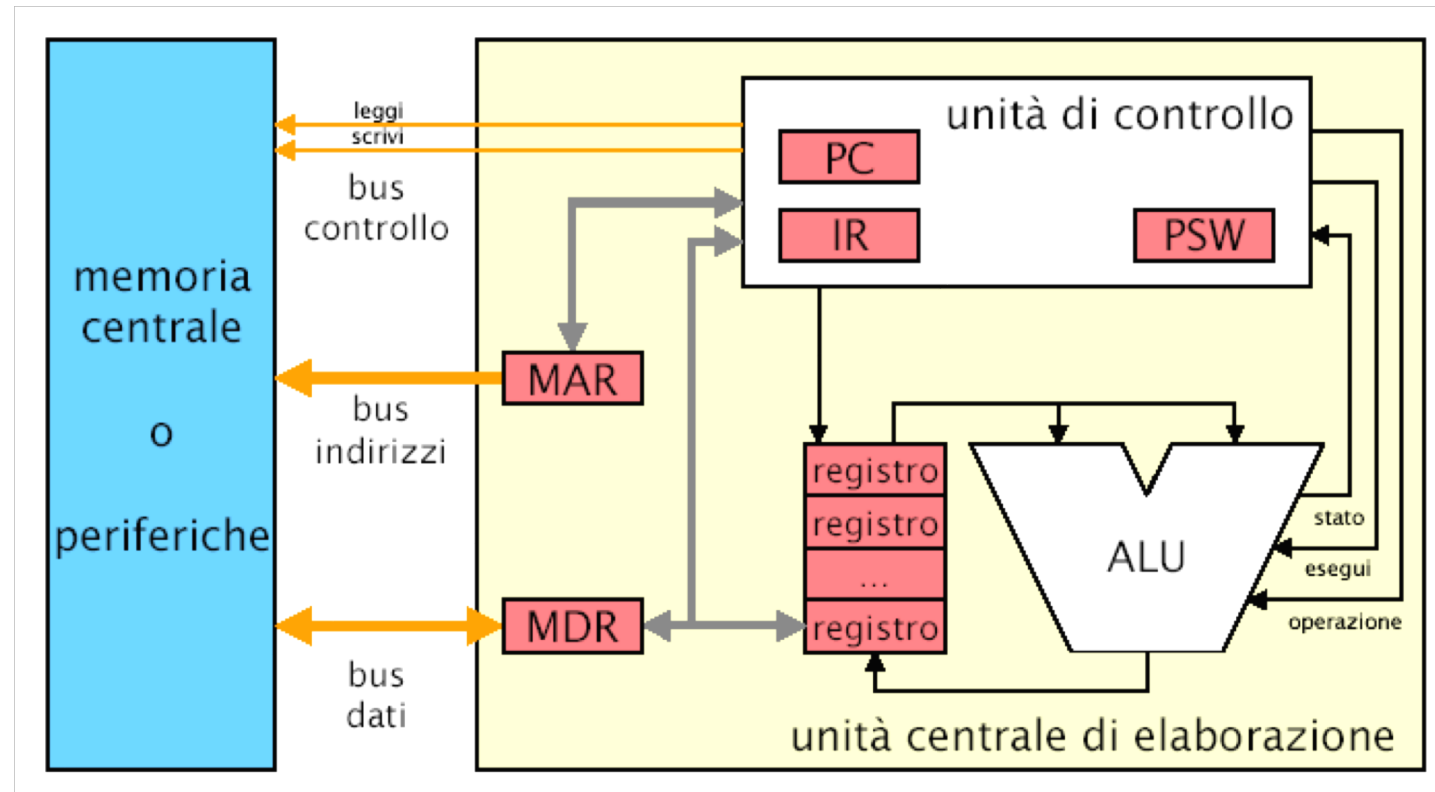
Architettura del Calcolatore



Architettura del Calcolatore

- La CPU esegue un'istruzione mediante le tre seguenti operazioni di base:
 - *Fetch* (lettura)
 - *Decode* (decodifica)
 - *Execute* (esecuzione)
- Un programma è eseguito applicando ad ogni istruzione la sequenza fetch-decode-execute, detta *ciclo di esecuzione dell'istruzione* o *ciclo macchina*

Architettura del Calcolatore



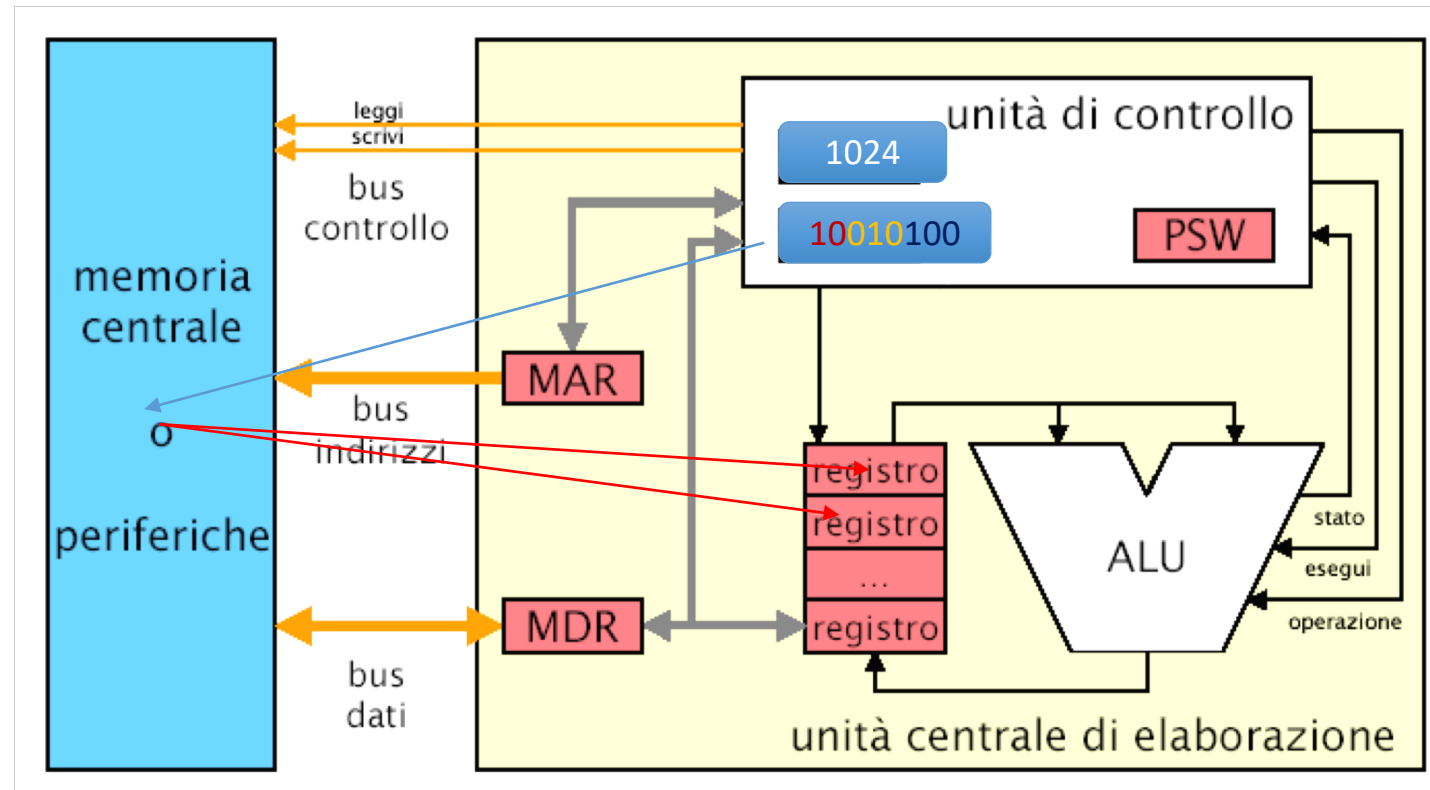
1) ***FETCH***:

-
- Il diagramma illustra l'architettura di un computer a istruzione singola (SISD). A sinistra, la **memoria centrale** (area blu) contiene dati e programmi, con un'indirizzo specifico **10010100** evidenziato. A destra, l'**unità centrale di elaborazione** (area gialla) è composta da:
- Unità di controllo**: Contiene un contatore di istruzioni (valore **1024**) e il **PSW** (Program Status Word).
 - Unità di elaborazione**: Include l'**ALU** (Arithmetic Logic Unit) e una serie di **registri**.
 - Registri**: Una pila di registri che memorizza dati e indirizzi.
- Le interazioni avvengono attraverso tre bus principali:
- bus controllo**: Collega l'unità di controllo alla memoria centrale.
 - bus indirizzi**: Collega l'unità di controllo al registro **MAR** (Memory Address Register).
 - bus dati**: Collega il registro **MDR** (Memory Data Register) alla memoria centrale.
- Il flusso di dati è indicato da frecce colorate: arancione per i bus, grigia per i trasferimenti interni all'unità centrale di elaborazione, e rossa per i trasferimenti diretti tra memoria e unità di elaborazione.

Architettura del Calcolatore

2) **DECODE**: decodifica dell'istruzione

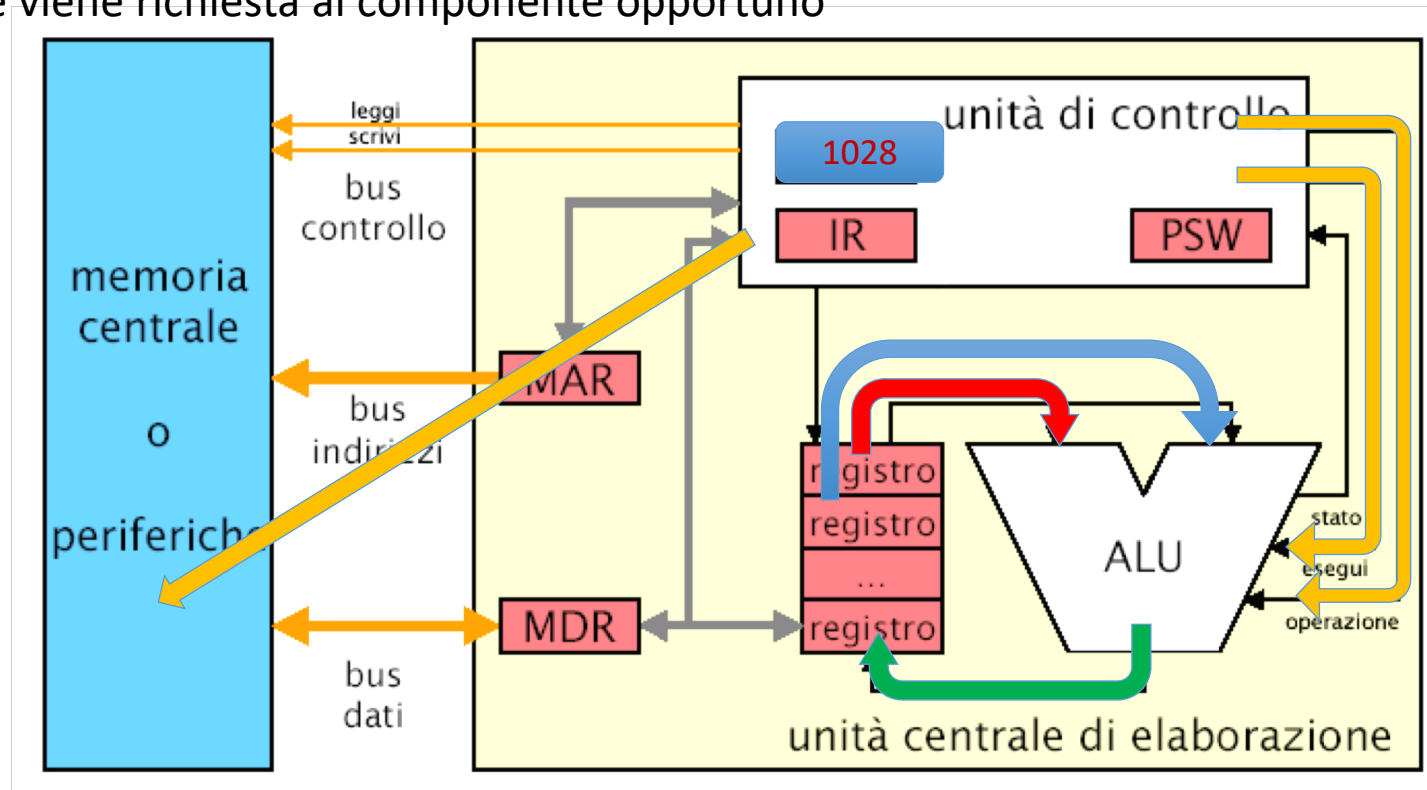
si individua il tipo dell'operazione e gli operandi (dati) usati si trasferiscono i dati nei registri opportuni



Architettura del Calcolatore

3) *EXECUTE*: esecuzione dell'istruzione

- si incrementa il registro contatore dell'istruzione (PC)
- ciascuna azione viene richiesta al componente opportuno

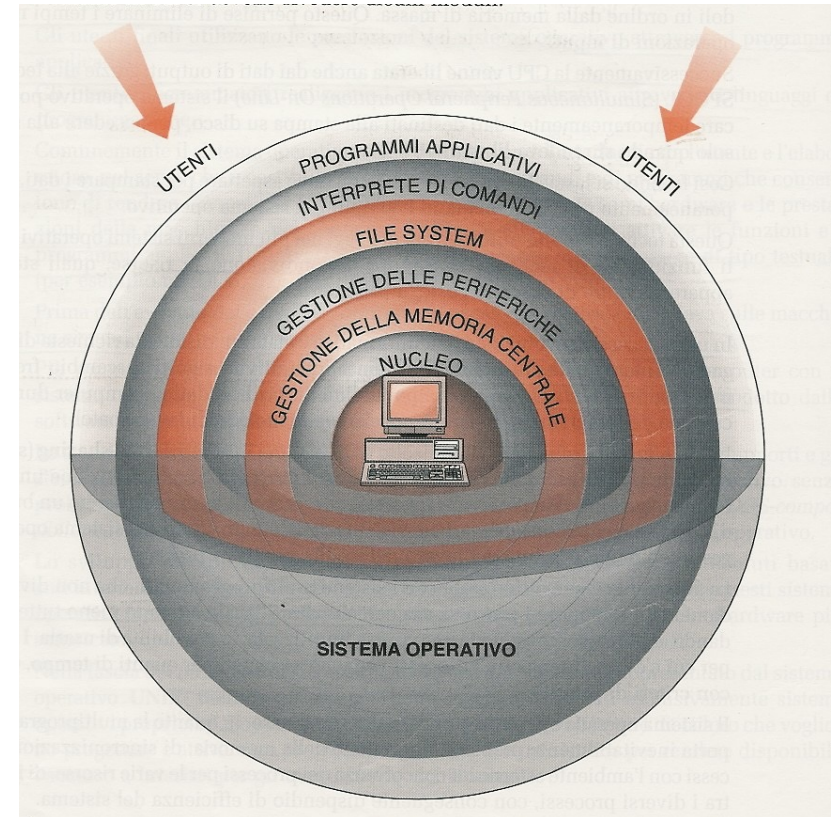


Sistema operativo

- Definizione:
 - Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore
- Obiettivi
 - Efficienza:
 - Un S.O. cerca di utilizzare in modo efficiente le risorse del calcolatore (***gestore delle risorse***), e quindi memoria, processore, periferiche.
 - Semplicità:
 - Un sistema operativo dovrebbe semplificare all'utente esterno l'uso dell'hardware di un calcolatore (***macchina astratta***)

Sistema operativo

- Il modello a cipolla rappresenta il sistema operativo come una successione di strati costruiti sopra la macchina hardware.
- Ciascuno degli strati della cipolla rappresenta un livello di macchina virtuale.
- In tutto, **sei** livelli di astrazione separano l'utente dall'hardware sottostante (incluso il livello utente)

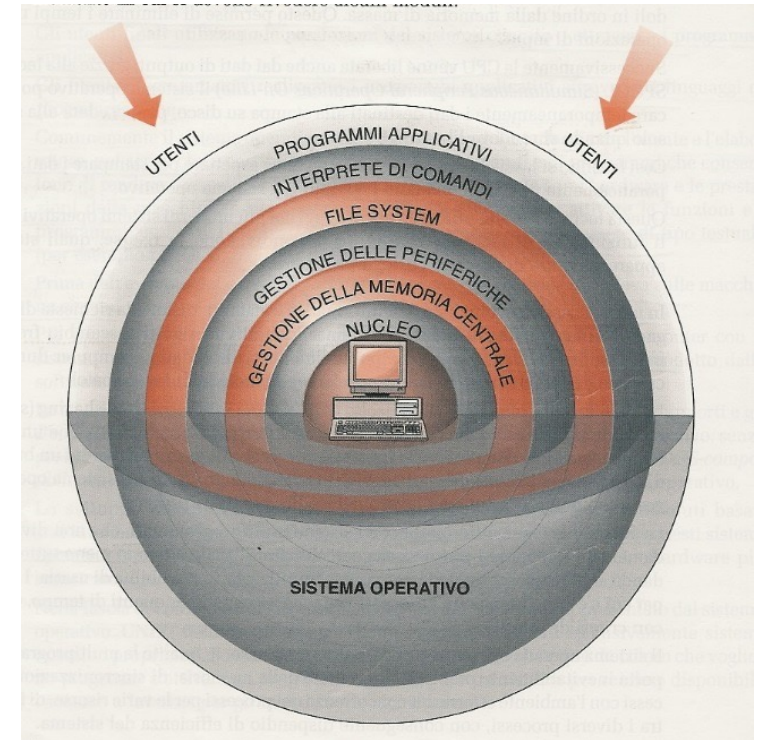


Sistema operativo

- Ogni strato (livello) costituisce una macchina virtuale:
 - Usa le funzionalità di quello sottostante
 - Fornisce servizi al livello superiore nella gerarchia
 - Gestisce delle risorse mediante politiche invisibili al livello sovrastante (struttura **modulare** del sistema operativo)

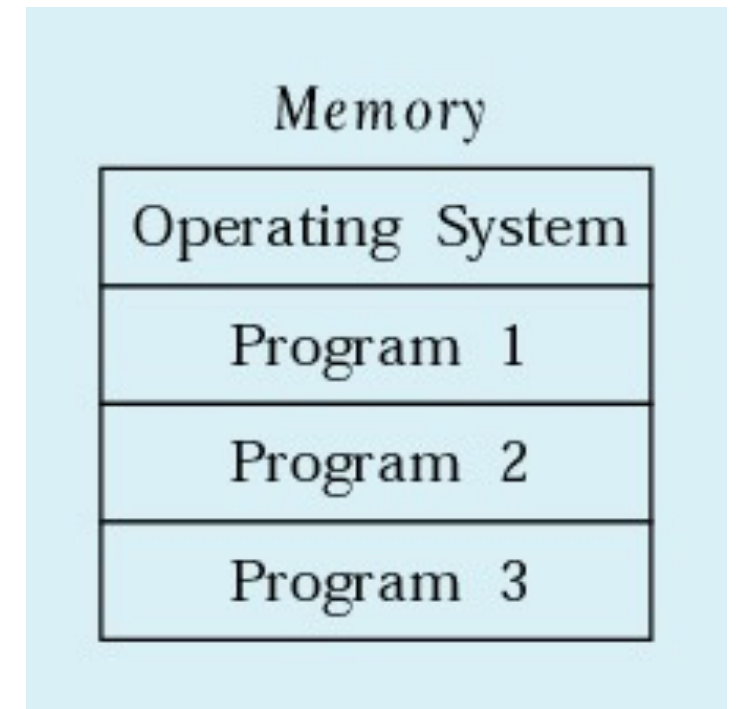
Sistema operativo

- Nucleo (kernel)
 - Gestisce l'elenco con priorità dei programmi pronti per l'esecuzione
 - Seleziona il programma prossimo da eseguire (prioritarizzazione)



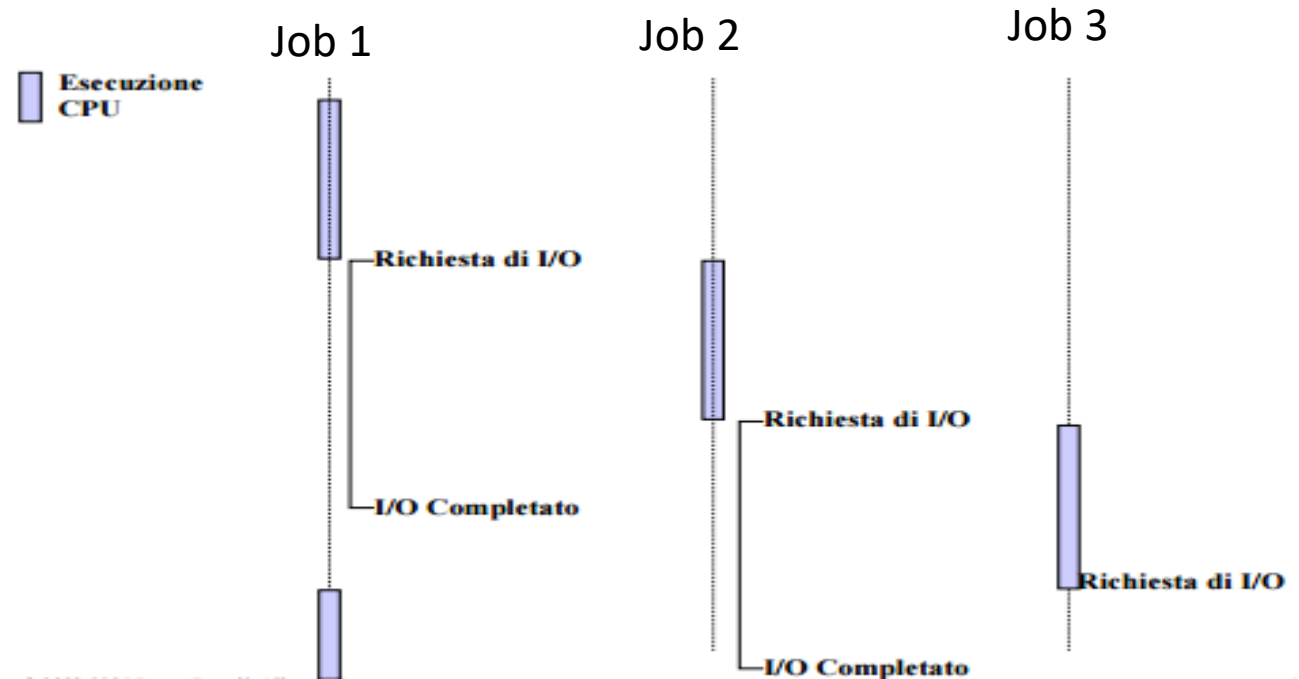
Sistema operativo

- **I Sistemi operativi erano inizialmente multiprogrammati** (gestione di più processi/jobs in memoria)
- Multiprogrammazione:
 - Utilizzare il processore per un altro job durante i periodi di I/O del job precedente
- Vantaggi:
 - Il processore non è lasciato inattivo (idle) durante attese molto lunghe
 - La memoria viene utilizzata al meglio, caricando il maggior numero di jobs possibili



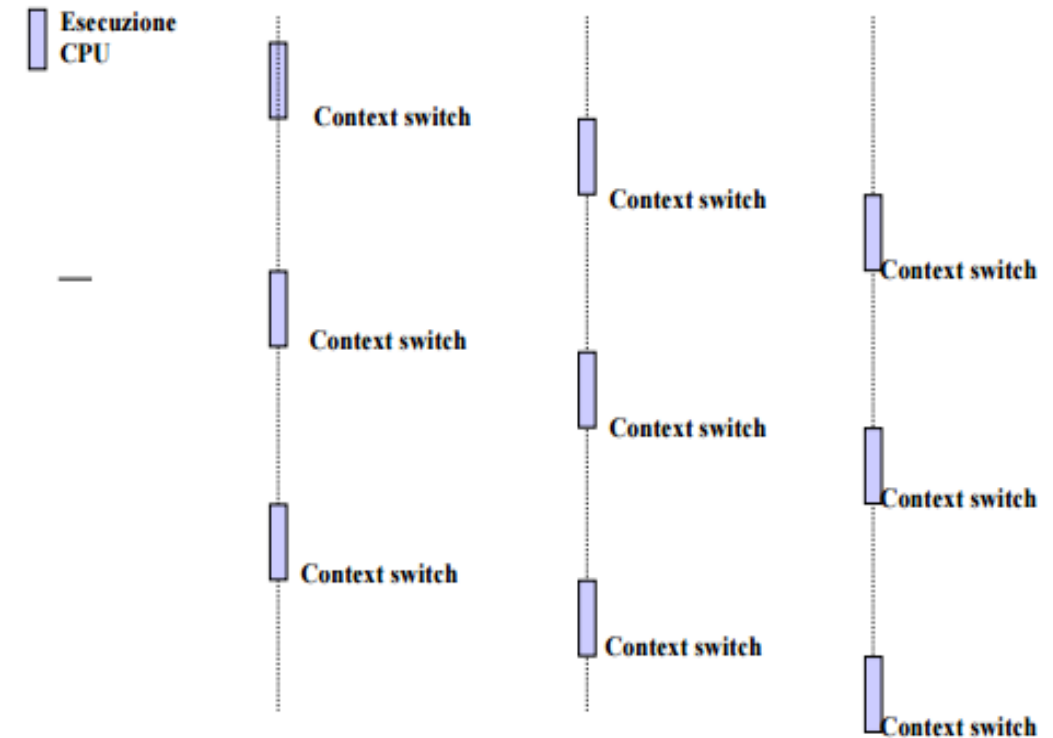
Sistema operativo

- Un componente del S.O. detto *scheduler* si preoccupa di alternare i job nell'uso della CPU
- Quando un job richiede una operazione di I/O, la CPU viene assegnata ad un altro job.



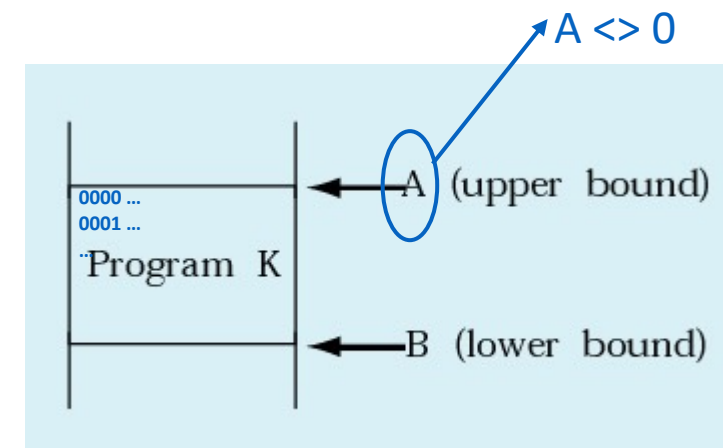
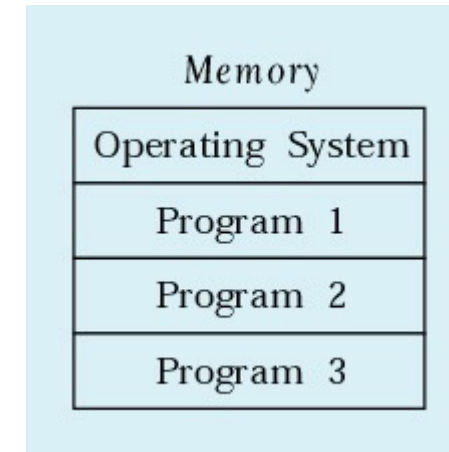
Sistema operativo

- La multiprogrammazione si evolve nel **Time sharing**
 - L'esecuzione della CPU viene suddivisa in quanti temporali
 - Allo scadere di un quanto, il job corrente viene interrotto e l'esecuzione passa ad un altro job
 - I passaggi (**context-switch**) avvengono così frequentemente che più utenti possono interagire con i programmi in esecuzione
 - Questo meccanismo consente la **multi-utenza**



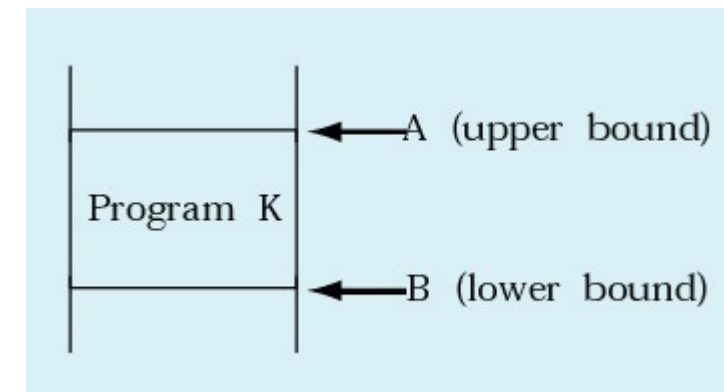
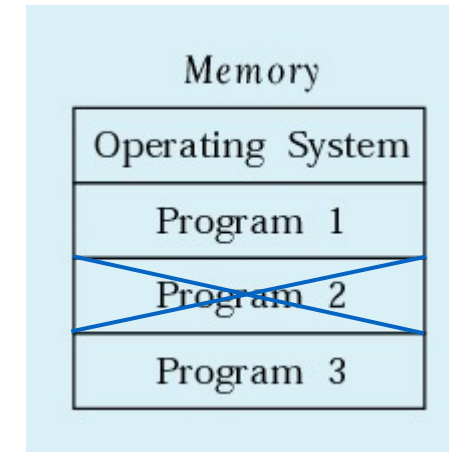
Sistema operativo

- Il Sistema operativo assegna un opportuno spazio di memoria ad ogni programma in esecuzione
- Il processore assume che ogni programma inizi dall'indirizzo 0
 - *Ma se un programma inizia dall'indirizzo 0 come faccio a garantire di poterlo caricare sempre all'indirizzo 0?*
 - Uso il punto di inizio per caricare correttamente il PC.
 - Codice **rilocabile**



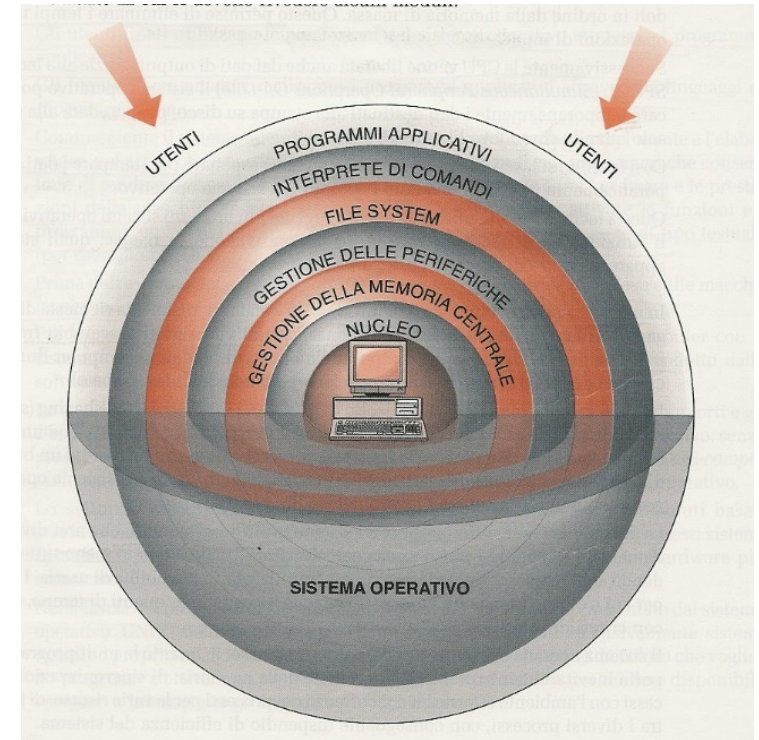
Sistema operativo

- *Che fare quando un programma termina e rilascia la sua porzione di memoria?*
 - Inserire un nuovo programma al posto del vecchio? E se non ci entra?
 - Lasciar perdere?



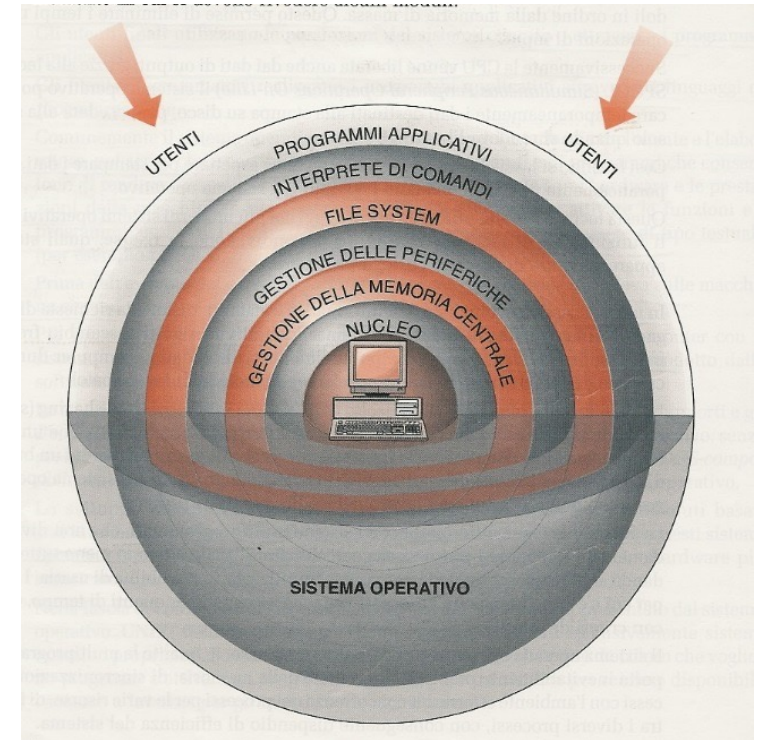
Sistema operativo

- Gestore della memoria
 - Riserva spazio in memoria per dati e programmi
 - Carica in memoria i programmi prima dell'esecuzione
- Gestori delle periferiche
 - Gestiscono la comunicazione con i dispositivi utilizzando Forniscono un'interfaccia software integrata del dispositivo (driver)



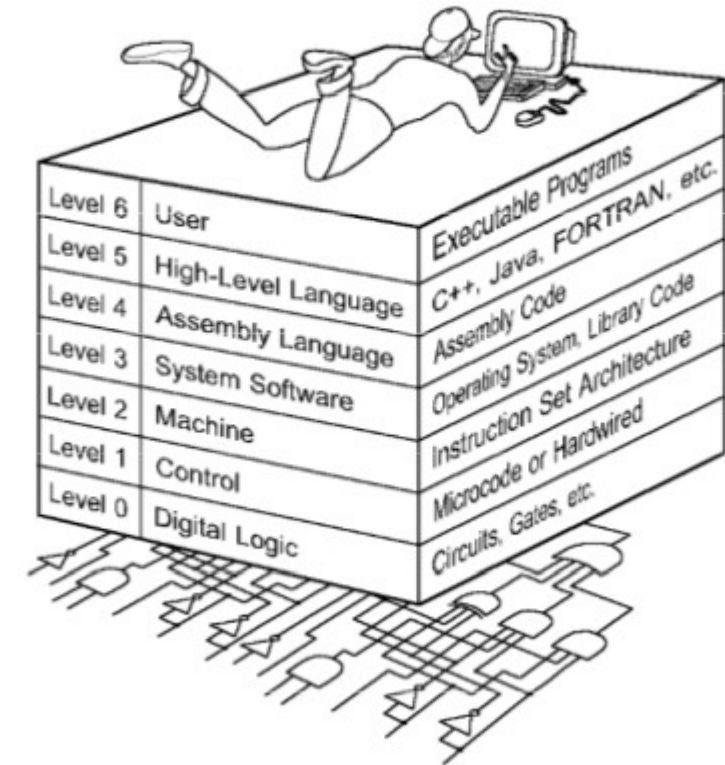
Sistema operativo

- File system
 - Gestisce la memorizzazione e il recupero di informazioni sui dispositivi di memoria di massa
- Interprete dei comandi
 - Esegue i comandi degli utenti e/o dei programmi applicativi
 - Interfaccia utente
 - Sicurezza negli accessi



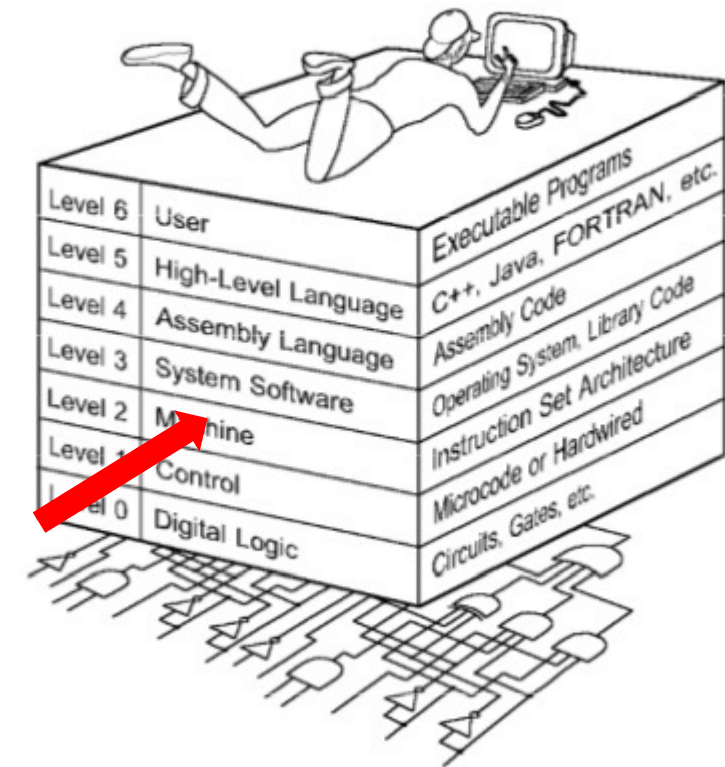
Linguaggi di Programmazione

- Sei livelli di astrazione separano l'utente dall'hardware sottostante
- Circuiti digitali
- Logica di controllo
- Macchina hardware
 - ISA – Instruction Set Architecture
- Software di base
 - Sistema operativo
 - librerie
- Linguaggio assemblativo
- **Linguaggi di programmazione**
- Programmi applicativi



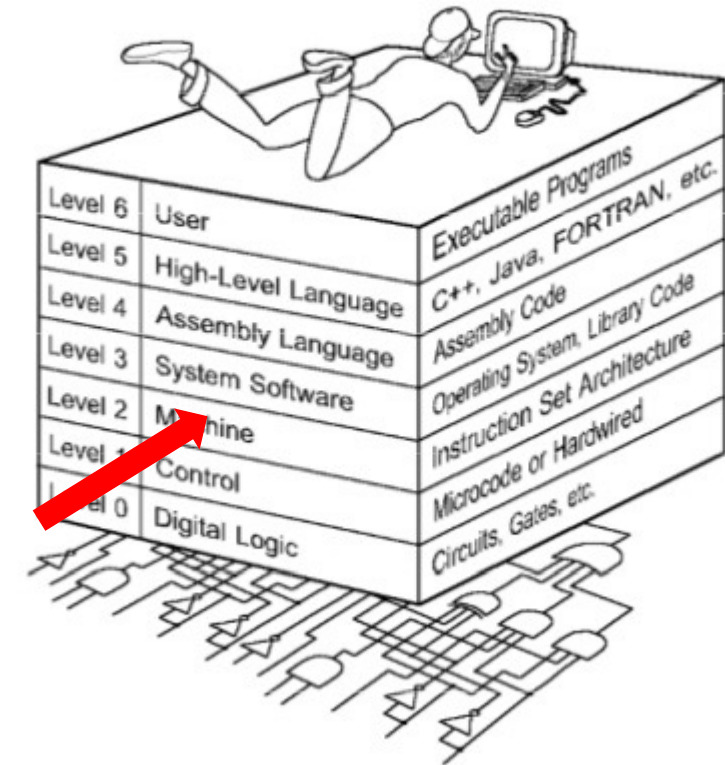
Linguaggi di Programmazione

- Linguaggio macchina
 - Formato binario. Le istruzioni sono indistinguibili dai dati su cui operano
 - Non consente l'uso di etichette o simboli per indicare locazioni di memoria o istruzioni adibite a compiti specifici
 - Difficile da modificare. Gli indirizzi delle istruzioni si susseguono sequenzialmente a partire dalla prima.
 - Difficile creare dati. I dati possono solo essere rappresentati nel loro formato interno



Linguaggi di Programmazione

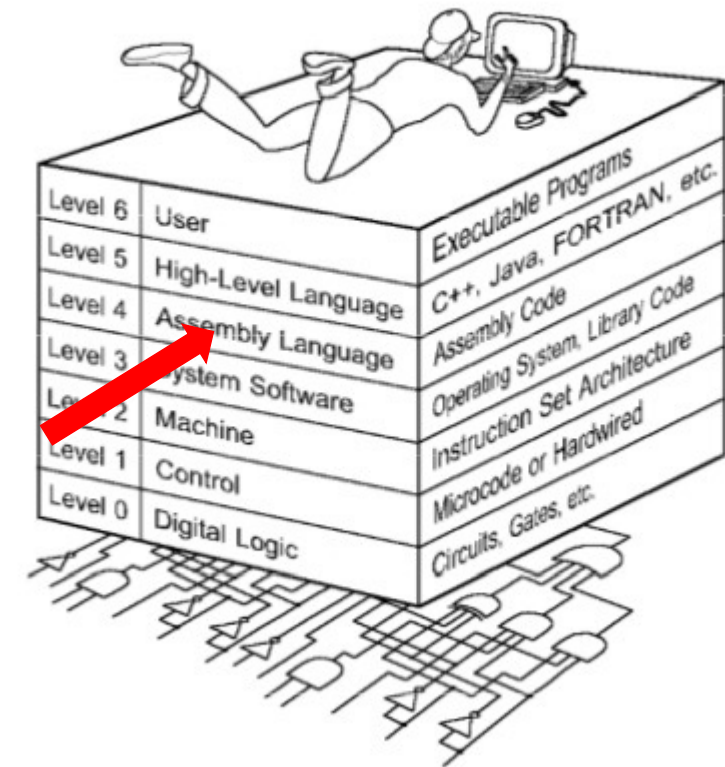
- Linguaggio macchina
 - I calcolatori della prima generazione potevano essere programmati soltanto in linguaggio macchina!
 - L'insieme delle istruzioni del linguaggio macchina di un processore firmano l'**I**nstruction **S**et **A**rchitecture (ISA)
 - Un processore **è** il suo ISA
 - Processori compatibili



Linguaggi di Programmazione

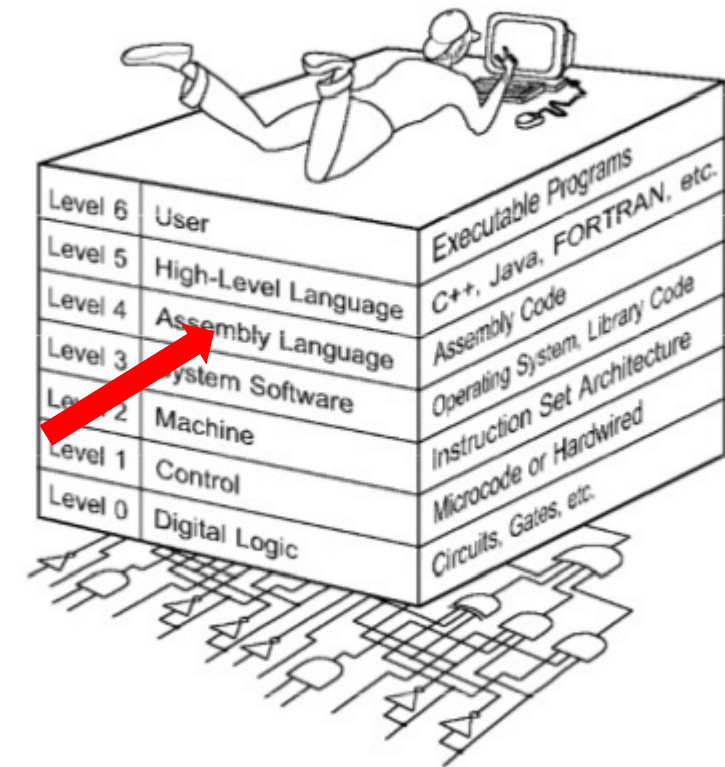
- Linguaggio assemblativo

- Orientato sia alla macchina che all'utente
- Le istruzioni sono indicate con etichette comprensibili che vengono tradotte nel codice binario corrispondente da un software *traduttore*
- Codici mnemonici:
 - ADD – addizione
 - SUB – sottrazione
 - LOAD, STORE – carica da memoria, memorizza in memoria
 - JUMP – salta ad istruzione successiva



Linguaggi di Programmazione

- Linguaggio assemblativo
- Rapporto 1:1 con il linguaggio macchina
 - Ogni istruzione in linguaggio assembler è tradotta esattamente nella sua corrispondente in linguaggio macchina
 - Specifico per una particolare classe di microprocessori



Linguaggi di Programmazione

Il primo
indirizzo di un
file oggetto è
sempre 0

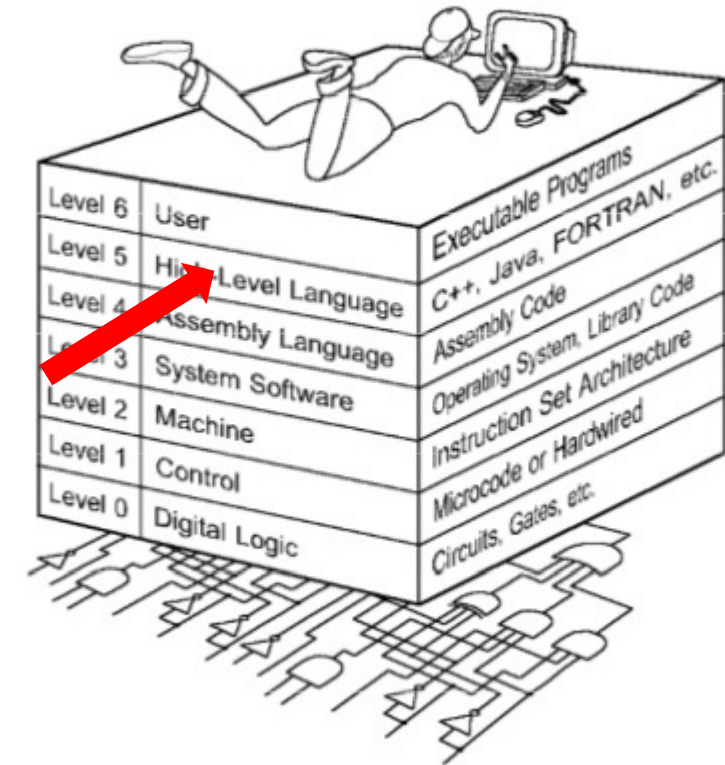
Indirizzo	Opcode data	Significato
0000	1101 000000001001	IN X
0001	1101 000000001010	IN Y
0010	0000 000000001001	LOAD X
0011	0111 000000001010	COMPARE Y
0100	1001 00000000111	JUMPGT DONE
0101	1110 000000001001	OUT X
0110	1000 000000000000	JUMP LOOP
0111	1110 000000001010	OUT Y
1000	1111 000000000000	HALT
1001	0000 000000000000	CONST 0
1010	0000 000000000000	CONST 0

Linguaggio Macchina

Linguaggio
Assemblativo

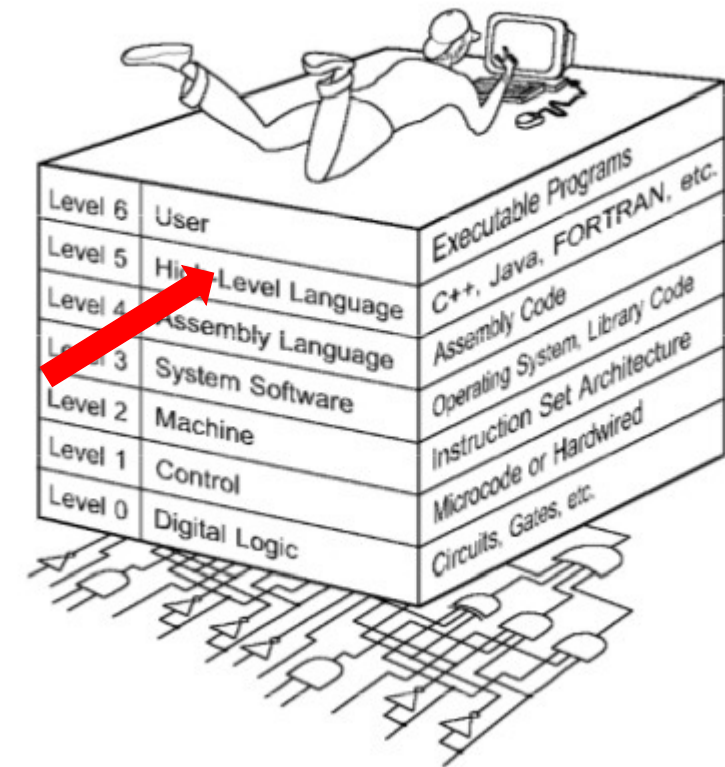
Linguaggi di Programmazione

- Linguaggi di programmazione di alto livello
 - Grazie alla gerarchia del software è possibile scrivere programmi in un linguaggio più vicino all'utente del linguaggio macchina
 - L'utente è ancora un programmatore esperto, profondo conoscitore della macchina e del suo funzionamento
 - Gestione automatica del movimento dei dati
 - Visione macroscopica dell'algoritmo, molto più vicina al pseudocodice



Linguaggi di Programmazione

- Linguaggi di programmazione di alto livello
 - Uso di operazioni primitive più potenti, che possono essere combinate per costruirne altre
 - Portabilità dei programmi da una macchina ad un'altra
 - Le istruzioni sono vicine al linguaggio naturale e usano la notazione matematica standard



Traduzione dei programmi

- Il programma scritto in linguaggio di alto livello (codice sorgente) dev'essere tradotto in linguaggio macchina da un apposito software detto **traduttore**

```
#include <stdio.h>
#define CONTA 3 // definisco CONTA con valore 3

int main(void)
{
    float n, media;
    int sum = 0; // calcolo il totale delle temperature
    int i;

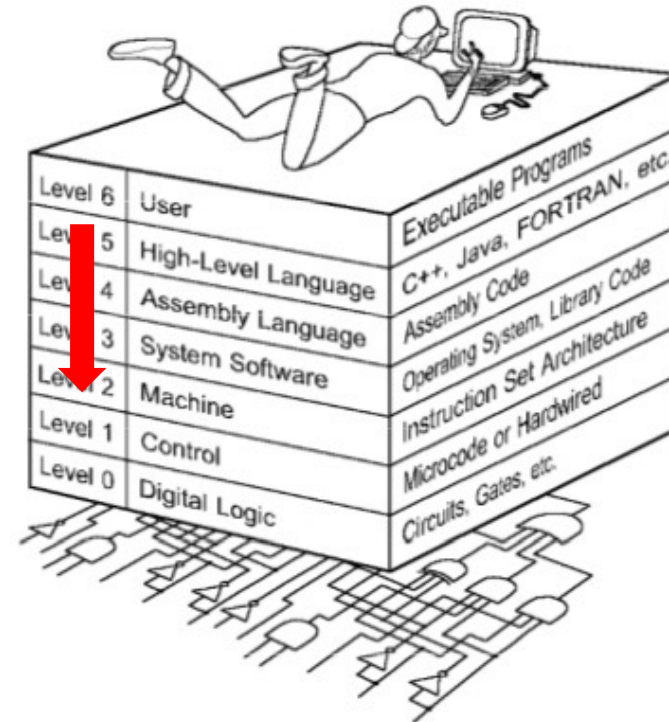
    // Supponiamo di poter inserire un massimo di 3 tem
    for (i=0; i<CONTA; i++){
        printf("Inserisci una temperatura: ");
        scanf("%f", &n);
        sum += n;
    }

    media = sum / CONTA;
    printf("\nLa media delle temperature e': %.2f", med

    return 0;
}
```



```
01010001 01110101 01100101 01110011 01110100
00100000 01110110 01101111 01101100 01110100
00100000 01101001 01101100 00100000 01110000
01110011 01110100 00100000 01101100 01101111
01110011 01100011 01110010 01101001 01110110
00100000 01101001 01101110 00100000 01100010
01101110 01100001 01110010 01101001 01101111
00001101 00001010 01000001 00100000 01100010
01101111 01101110 00100000 01100011 01101111
```



Traduzione dei programmi

- Compilatori

- Effettuano la traduzione del programma dal linguaggio di programmazione sorgente al codice macchina, generando un programma eseguibile che permane, dopo la traduzione, come file a se stante.
- Programmi più efficienti.
- Necessitano una nuova traduzione in caso di errori run-time.

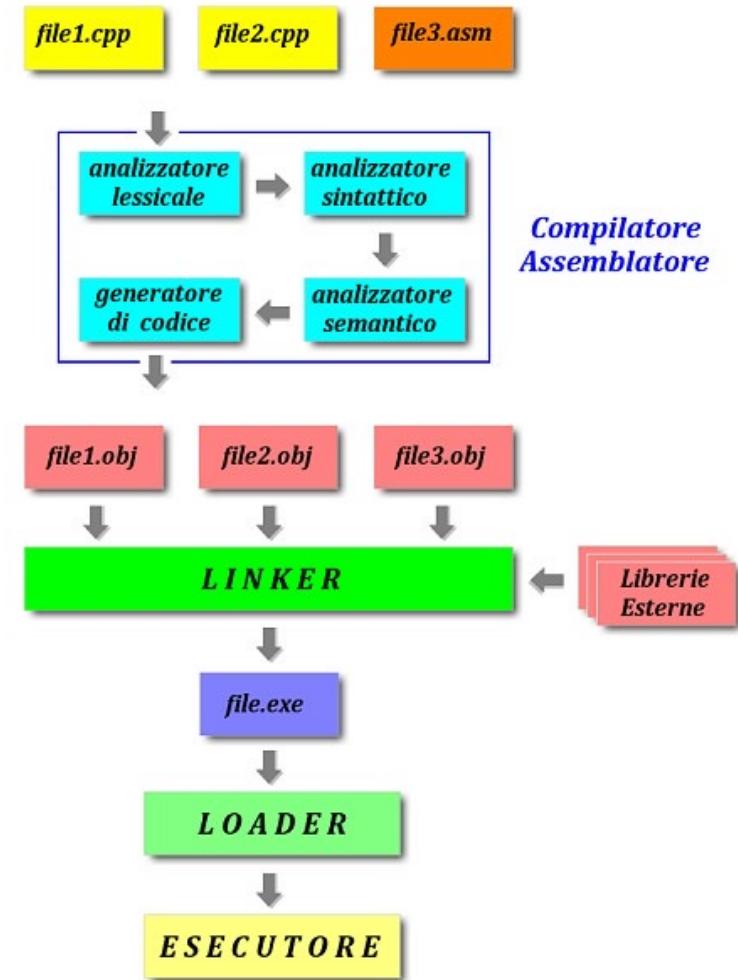
- Interpreti

- Effettuano la traduzione istruzione per istruzione del programma sorgente, generando il codice macchina passo passo, senza creare un programma eseguibile statico.
- Programmi meno efficienti.
- Correzione interattiva degli errori.

Traduzione dei programmi

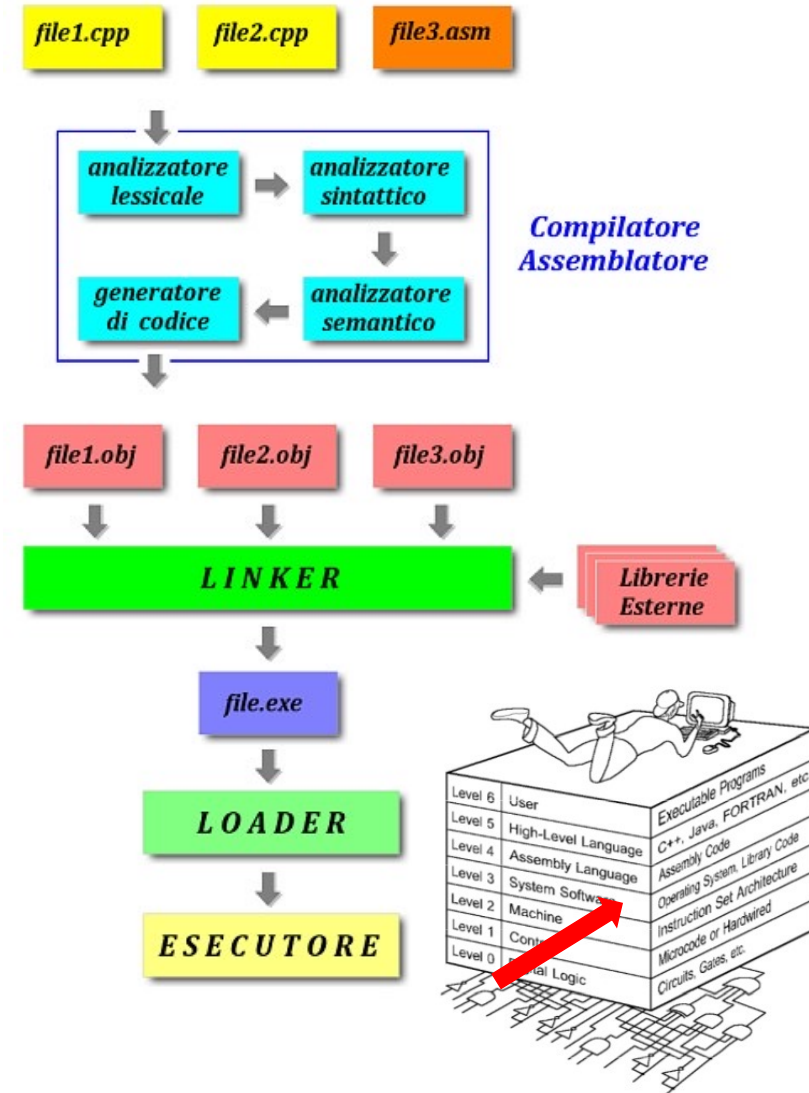
- Compilatore

- Trasforma il programma sorgente in codice oggetto
- 4 fasi di lavoro
 - Analisi delle stringhe di testo (token)
 - Analisi della correttezza sintattica delle frasi
 - Analisi della correttezza “semantica” cioè della coerenza rispetto alle altre frasi
 - Generazione del codice binario
- Include e/o richiama l’assemblatore



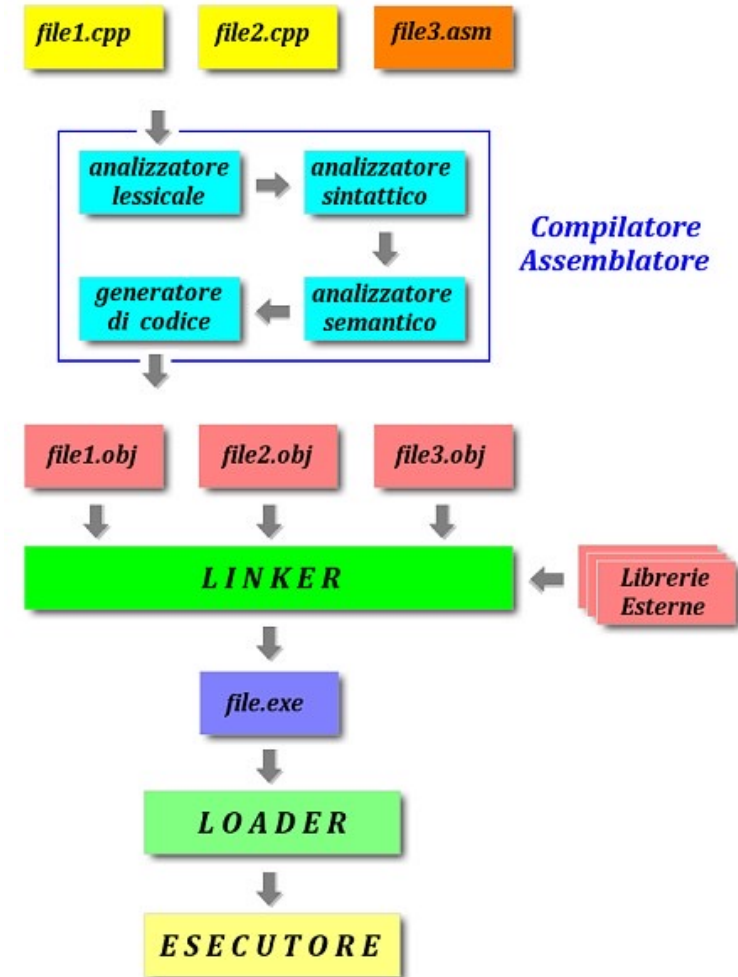
Traduzione dei programmi

- Assemblatore
 - Trasforma direttamente il codice assembleativo in linguaggio macchina
- Linker
 - Collega al codice oggetto del programma altri frammenti di codice oggetto (**librerie**)
 - Le librerie sono i codici di accesso alle funzioni del Sistema Operativo
- Le librerie rappresentano il passo di traduzione operato dal livello 3 della gerarchia



Traduzione dei programmi

- Loader
 - Non è un passo di traduzione
 - Carica il file eseguibile in memoria centrale per l'esecuzione
 - Rilocalizzazione del codice
 - Gestione della memoria



Traduzione dei programmi

- Le quattro fasi vengono realizzate istruzione per istruzione
- Non c'è il loader perché ogni istruzione tradotta è eseguita direttamente
- Le librerie utilizzate sono le cosiddette **run-time**
 - Pronte per essere eseguite direttamente e non da collegare ad altri codici oggetto

